In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
df_full=pd.read_csv("DataB.csv")
```

In [3]:
```
df_full
```

Out[3]:

	Unnamed: 0	fea.1	fea.2	fea.3	fea.4	fea.5	fea.6	fea.7	fea.8	fea.9	...	fea.776	fea.777	f
0	1	4	4	3	0	0	4	2	1	4	...	1	3	
1	2	5	1	4	3	1	3	5	1	4	...	1	1	
2	3	1	3	0	3	1	1	0	1	0	...	3	0	
3	4	5	3	2	3	5	2	2	0	4	...	5	4	
4	5	3	5	3	3	0	4	1	1	4	...	1	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2061	2062	4	0	3	0	4	0	4	3	1	...	0	1	
2062	2063	2	2	3	4	2	1	2	3	3	...	4	0	
2063	2064	2	3	2	3	1	2	5	5	5	...	5	1	
2064	2065	5	2	4	3	1	0	3	2	2	...	3	2	
2065	2066	3	3	1	3	2	5	4	2	2	...	2	3	

2066 rows × 786 columns

In [4]:
```
df_features=df_full.filter(regex=("fea.*"))
```

In [5]:
```
df_features
```

Out[5]:
```
      fea.1 fea.2 fea.3 fea.4 fea.5 fea.6 fea.7 fea.8 fea.9 fea.10 ... fea.775 fea.776 fea.7
   0      4     4     3     0     0     4     2     1     4      1 ...       1       1
   1      5     1     4     3     1     3     5     1     4      4 ...       3       1
   2      1     3     0     3     1     1     0     1     0      2 ...       4       3
   3      5     3     2     3     5     2     2     0     4      5 ...       4       5
   4      3     5     3     3     0     4     1     1     4      3 ...       1       1
 ...    ...   ...   ...   ...   ...   ...   ...   ...   ...    ... ...     ...     ...
2061      4     0     3     0     4     0     4     3     1      2 ...       3       0
2062      2     2     3     4     2     1     2     3     3      4 ...       1       4
2063      2     3     2     3     1     2     5     5     5      0 ...       3       5
2064      5     2     4     3     1     0     3     2     2      1 ...       2       3
2065      3     3     1     3     2     5     4     2     2      4 ...       3       2

2066 rows × 784 columns
```

In [6]:
```
# df_features=StandardScaler().fit_transform(df_fea) # applying normalizati
on
# df_features=((df_fea-df_fea.mean())) #mean centered only .
```

In [7]:
```
from scipy.stats import zscore #taking zscore as mentioned on piazza. pca r
equires mean centered data.
df_features_znorm=df_features.apply(zscore)
```

```
In [8]: df_features_znorm
```

Out[8]:

	fea.1	fea.2	fea.3	fea.4	fea.5	fea.6	fea.7	fea.8	
0	1.010077	0.966782	0.359594	-1.668004	-1.638671	1.007994	-0.324102	-0.992537	0.9
1	1.687176	-1.029924	1.026488	0.336317	-0.976018	0.340307	1.674690	-0.992537	0.9
2	-1.021220	0.301213	-1.641090	0.336317	-0.976018	-0.995067	-1.656630	-0.992537	-1.6
3	1.687176	0.301213	-0.307301	0.336317	1.674594	-0.327380	-0.324102	-1.648724	0.9
4	0.332978	1.632350	0.359594	0.336317	-1.638671	1.007994	-0.990366	-0.992537	0.9
...	...	...	...	...	...	...	...	...	
2061	1.010077	-1.695492	0.359594	-1.668004	1.011941	-1.662753	1.008426	0.319835	-1.0
2062	-0.344121	-0.364355	0.359594	1.004424	-0.313365	-0.995067	-0.324102	0.319835	0.3
2063	-0.344121	0.301213	-0.307301	0.336317	-0.976018	-0.327380	1.674690	1.632208	1.6
2064	1.687176	-0.364355	1.026488	0.336317	-0.976018	-1.662753	0.342162	-0.336351	-0.3
2065	0.332978	0.301213	-0.974195	0.336317	-0.313365	1.675681	1.008426	-0.336351	-0.3

2066 rows × 784 columns

1. In PCA, compute the eigenvectors and eigenvalues. Plot the scree plot and visually discuss which cut-off is good.

*finding covariance matrix --> (xt.x). This will lead to d*d matrix along dimensions .*

```
In [9]: covariance_matrix=np.cov(df_features_znorm.T) #.T otherwise use rowvar=Fals
        e. if true each row reperesnts a variable.
```

```
In [10]: covariance_matrix=pd.DataFrame(covariance_matrix)
```

In [11]:
```
covariance_matrix.head() # here 0 stands for fea1 and so on.calculated on f
eature space xt.x .. 0 -->i.e feature 1's  variance with d features in firs
t column and so on ..
```

Out[11]:

	0	1	2	3	4	5	6	7	
0	1.000484	-0.020254	0.028026	-0.027699	-0.003159	0.026745	-0.007159	-0.027586	-0.0077
1	-0.020254	1.000484	0.021676	-0.033306	-0.008296	0.047226	0.001896	0.003229	-0.0355
2	0.028026	0.021676	1.000484	0.000480	-0.022512	-0.009226	-0.016804	-0.005930	0.0165
3	-0.027699	-0.033306	0.000480	1.000484	0.008106	-0.009736	-0.030415	0.009572	0.0073
4	-0.003159	-0.008296	-0.022512	0.008106	1.000484	-0.047146	-0.050727	0.010993	0.0005

5 rows × 784 columns

finding the eigen vectors and corresponding eigen values and sorting them out in decreasing order.

In [12]:
```
values, vectors = np.linalg.eigh(covariance_matrix)  #eigh? changing the si
gns. because its a symmetric cov matrix therefore eigh
```

In [13]:
```
sorted_values = np.argsort(values)[::-1] #sorting acc to index reverse retu
rns array of indices of same shape.
```

In [14]:
```
# sorted_values
```

In [15]:
```
# sorted_values=pd.DataFrame(sorted_values)
# sorted_values.head()
# sorted_values=sorted_values.to_numpy()
```

In [16]:
```
sorted_vectors = vectors[:,sorted_values] #sorting the vectors acc to same
 index
```

In [17]:
```
values = values[sorted_values] #sorting values
```

In [18]:
```
values=pd.DataFrame(values)
```

In [19]:
```
sorted_vectors=pd.DataFrame(sorted_vectors.T)
```

eigen values: the amount of variance explained by each of these vectors/axes/components.

In [20]:
```
values
```

Out[20]:

	0
0	51.777319
1	28.800865
2	26.770911
3	23.930346
4	21.575039
...	...
779	0.008340
780	0.008145
781	0.007711
782	0.007438
783	0.007105

784 rows × 1 columns

Directions created with highest variance with respect to data in feature space. i.e first column highest e value will lead to first pc when projected with original data. eigen vectors: pricipal axes in feature space representing the maximum variance in data.

In [21]:
```
sorted_vectors
```

Out[21]:

	0	1	2	3	4	5	6	7	
0	0.001979	0.001513	-0.000492	-0.003617	-0.000810	0.005896	-0.001439	-0.001285	-0.00
1	0.004933	-0.006404	-0.001566	0.000518	-0.002937	-0.003685	-0.008790	0.000596	-0.00
2	-0.000375	0.002587	-0.003725	0.000352	-0.001220	-0.000092	-0.002634	0.005623	-0.00
3	0.006154	0.002790	-0.003226	-0.002489	0.010679	0.000604	-0.001788	-0.003557	-0.00
4	-0.003251	0.003767	0.001384	-0.000363	-0.001331	0.008793	0.002887	0.007900	-0.00
...	...	...	...	...	...	...	...	...	
779	0.000990	0.002825	0.002969	0.007797	0.005491	0.001475	0.001826	0.004143	-0.00
780	-0.003076	-0.002147	0.000022	0.002503	0.006257	-0.003685	0.000386	0.003653	-0.00
781	0.002942	0.001887	0.008383	-0.001179	-0.004695	-0.002205	-0.002482	-0.000407	-0.00
782	-0.000692	0.004045	0.000907	0.004380	0.003882	-0.000384	-0.000257	0.000715	-0.00
783	0.001785	0.004795	-0.008164	0.004050	-0.004779	-0.001970	-0.007182	0.003580	0.00

784 rows × 784 columns

In [22]:
```
# # can also be done simply by::::::::::::::::::::

# pca=PCA()
# pca=pca.fit(df_features_znorm)  #not transform, it will make pc's
# pca_eigen_vectors=pca.components_ #defining the direction of the vector
# pca_eigen_vectors_df=pd.DataFrame(pca_eigen_vectors)
# # pca_eigen_vectors_df
# pca_eigen_values=pca.explained_variance_ #defining the length of the vector
# pca_eigen_values_df=pd.DataFrame(pca_eigen_values)
# pca_eigen_vectors_df
# pca_exp_var_ratio=pca.explained_variance_ratio_
#pca_exp_var_ratio=pd.DataFrame(pca_exp_var_ratio)
# pca_exp_var_ratio
```

Percentage of variance explained by each of the selected components. Sum is 1 if all the components are taken.

In [23]:
```
# count_values=df_features_znorm.shape[0]
sum_values=values.sum()
explained_variance_proportion_variation_around_origin_for_each_pc=values/sum_values
```

In [24]:
```
explained_variance_proportion_variation_around_origin_for_each_pc[0:15]
```

Out[24]:

	0
0	0.066011
1	0.036718
2	0.034130
3	0.030509
4	0.027506
5	0.020263
6	0.017672
7	0.015187
8	0.013592
9	0.012524
10	0.011619
11	0.010566
12	0.009524
13	0.009162
14	0.008947

Cumulative explained variance

In [25]:
```
cum_exp_variance = np.cumsum(explained_variance_proportion_variation_around
_origin_for_each_pc.to_numpy())
```

In [26]:
```
# cum_exp_variance=pd.DataFrame(cum_exp_variance)
```

In [27]:

```
cum_exp_variance*100
```

```
Out[27]: array([ 6.60105329, 10.27285485, 13.68585902, 16.73672168,
                19.48730796, 21.51356207, 23.28081102, 24.79947466,
                26.15869123, 27.41109276, 28.57294545, 29.62957132,
                30.58194441, 31.49819163, 32.39293912, 33.25084497,
                34.03767231, 34.80647794, 35.56786792, 36.30040669,
                37.01006555, 37.69096736, 38.36793512, 39.01461745,
                39.6395954 , 40.24393637, 40.81259078, 41.3784849 ,
                41.9362798 , 42.4780984 , 43.013217  , 43.53999149,
                44.05080072, 44.54795772, 45.03509904, 45.51420603,
                45.98602757, 46.44147372, 46.89262097, 47.33578555,
                47.76652109, 48.18432108, 48.5917961 , 48.99818359,
                49.39337824, 49.77666856, 50.153614  , 50.52793941,
                50.89085986, 51.2484551 , 51.60255969, 51.95132732,
                52.29945225, 52.63606688, 52.96784744, 53.29492938,
                53.61146856, 53.92578902, 54.23668959, 54.54399805,
                54.84819088, 55.1470685 , 55.44169147, 55.72964665,
                56.01394994, 56.2942943 , 56.56888597, 56.84244417,
                57.10930867, 57.37234371, 57.6340219 , 57.8897467 ,
                58.14085033, 58.39029526, 58.63584891, 58.87860541,
                59.11837117, 59.3533717 , 59.58701097, 59.81858321,
                60.04920481, 60.27873367, 60.50712238, 60.73382614,
                60.95737106, 61.17973867, 61.40059536, 61.61949857,
                61.83567062, 62.04941948, 62.26258793, 62.47474757,
                62.6854854 , 62.89534271, 63.10271449, 63.3080022 ,
                63.51163423, 63.71475494, 63.91640257, 64.11633661,
                64.31495681, 64.51339294, 64.71116292, 64.90683161,
                65.10093512, 65.29452018, 65.48642558, 65.67817339,
                65.86919425, 66.05926057, 66.24808191, 66.43637397,
                66.62396686, 66.81092352, 66.99573861, 67.17996666,
                67.36355342, 67.54548714, 67.72684508, 67.90738935,
                68.08693536, 68.26570609, 68.44364304, 68.62069975,
                68.79651357, 68.97218183, 69.14630713, 69.32016691,
                69.49305318, 69.66489828, 69.83559318, 70.00516892,
                70.17447941, 70.3431055 , 70.51129696, 70.67787884,
                70.84409704, 71.00970677, 71.17506123, 71.33981609,
                71.50340702, 71.66642701, 71.82873158, 71.99007694,
                72.15046509, 72.31037254, 72.46982151, 72.62860895,
                72.78680141, 72.94371053, 73.10039769, 73.25624864,
                73.41139072, 73.5656817 , 73.71922896, 73.87182627,
                74.02432259, 74.17598052, 74.32699523, 74.47733797,
                74.62732019, 74.77610115, 74.92408499, 75.07176306,
                75.21888068, 75.36573554, 75.51156323, 75.65698922,
                75.80201126, 75.94631585, 76.09055714, 76.23388147,
                76.37644994, 76.51866267, 76.66030219, 76.80071755,
                76.94056172, 77.07951527, 77.21789477, 77.3560971 ,
                77.49324141, 77.62995958, 77.7664295 , 77.90201244,
                78.03704412, 78.17149783, 78.30569853, 78.43943957,
                78.5727319 , 78.70563226, 78.83764962, 78.96903917,
                79.10008971, 79.23089921, 79.36101824, 79.49096404,
                79.62023763, 79.74850166, 79.87603547, 80.00317302,
                80.13019061, 80.25672409, 80.38282987, 80.50874766,
                80.63437125, 80.75957691, 80.88369261, 81.00741901,
                81.13089893, 81.25355114, 81.37584887, 81.49756821,
                81.61854326, 81.73929474, 81.85974228, 81.97978019,
                82.0991381 , 82.21815286, 82.33635244, 82.45353253,
                82.57022565, 82.68619997, 82.80176751, 82.91705833,
                83.03192255, 83.14614663, 83.26005107, 83.37315636,
                83.48613126,  83.59852591,  83.71014922,  83.82143834,
                83.93211258,  84.04258045,  84.15257371,  84.26180421,
                84.37095625,  84.47981092,  84.58806642,  84.69600288,
                84.80310753,  84.90918133,  85.01497162,  85.12068913,
                85.22585615,  85.3307343 ,  85.43526778,  85.53932369,
                85.64239709,  85.74532071,  85.84811792,  85.94982811,
                86.05150566,  86.15217888,  86.25267332,  86.35282237,
                86.45272946,  86.55208221,  86.6507616 ,  86.74852505,
                86.84608075,  86.94312568,  87.03949237,  87.13547534,
                87.23119328,  87.3267161 ,  87.42152985,  87.51587933,
                87.6097583 ,  87.70335205,  87.79638788,  87.88903046,
                87.98153079,  88.07349071,  88.16457981,  88.25563879,
                88.34610576,  88.43621138,  88.52592741,  88.61548241,
                88.70438268,  88.79257661,  88.88060897,  88.96820973,
                89.05531093,  89.14115364,  89.22683461,  89.312359  ,
                89.39745228,  89.48234758,  89.56694356,  89.65098848,
                89.73468336,  89.81803003,  89.90085491,  89.98315141,
                90.06495166,  90.14642704,  90.22730467,  90.30780586,
                90.38788732,  90.4679329 ,  90.54728262,  90.62632098,
                90.70482237,  90.78283624,  90.86022601,  90.93749271,
                91.01399125,  91.09013979,  91.16598747,  91.24150392,
                91.3167636 ,  91.39151304,  91.46563591,  91.53918182,
                91.61263027,  91.68565312,  91.7582556 ,  91.83035761,
                91.90185386,  91.97313837,  92.04397181,  92.114625  ,
                92.18421956,  92.25354049,  92.3226053 ,  92.39136694,
                92.4590618 ,  92.52664462,  92.59372881,  92.66039286,
                92.72683016,  92.7928936 ,  92.85870385,  92.92352891,
                92.98792764,  93.05190308,  93.11511316,  93.17807376,
                93.24017175,  93.30220416,  93.3636754 ,  93.42449389,
                93.48493486,  93.54470048,  93.60429891,  93.66313551,
                93.72161299,  93.77954362,  93.83735512,  93.89436789,
                93.95123364,  94.00782769,  94.06390695,  94.11953682,
                94.17454331,  94.22871222,  94.28279701,  94.33662711,
                94.38960971,  94.44220586,  94.49450629,  94.54652507,
                94.59735182,  94.64797985,  94.69769226,  94.74723965,
                94.79622545,  94.84470231,  94.89299281,  94.94100165,
                94.98850945,  95.03557409,  95.08212845,  95.12842492,
                95.17417206,  95.21959091,  95.26473558,  95.30924981,
                95.35351329,  95.39719838,  95.44038214,  95.48316839,
                95.52581524,  95.56775698,  95.60943007,  95.65057078,
                95.69137936,  95.73190526,  95.77195769,  95.81177266,
                95.85125827,  95.89026559,  95.92862198,  95.96663185,
                96.00424522,  96.04152529,  96.07847764,  96.11500179,
                96.15135477,  96.18729948,  96.22279359,  96.25808387,
                96.29320891,  96.32780888,  96.36184547,  96.39566219,
                96.42894834,  96.46203513,  96.49454932,  96.5265297 ,
                96.5583624 ,  96.58995804,  96.62131545,  96.6520862 ,
                96.68274495,  96.71309793,  96.74316967,  96.7729336 ,
                96.80245389,  96.83185642,  96.86104554,  96.88994249,
                96.91864697,  96.94684952,  96.97476789,  97.00239386,
                97.02986465,  97.05698155,  97.0839273 ,  97.11056379,
                97.13698938,  97.16325483,  97.18917218,  97.21462049,
                97.23992352,  97.26514933,  97.28986156,  97.31446401,
                97.3389009 ,  97.36293393,  97.3868129 ,  97.4104683 ,
                97.43386073,  97.45707539,  97.48014595,  97.50291964,
                97.5255678 ,  97.54796996,  97.57000918,  97.5919721 ,
                97.61381501,  97.63530352,  97.65664314,  97.67781748,
                97.69872585,  97.71950145,  97.74017151,  97.76081553,
                97.78106006,  97.80088895,  97.82065957,  97.84012506,
                97.85934183,  97.87850415,  97.89729365,  97.91601155,
                97.93451662,  97.95292609,  97.971221  ,  97.98931178,
                98.0072295 ,  98.0249871 ,  98.04260436,  98.05990556,
                98.07711585,  98.0942864 ,  98.11132128,  98.12808828,
                98.14480354,  98.16130487,  98.17775934,  98.19410999,
                98.21025073,  98.22628521,  98.24219202,  98.25795448,
                98.27339292,  98.28868041,  98.30394481,  98.31917679,
                98.33430012,  98.34927692,  98.36408204,  98.37881591,
                98.39335906,  98.40784631,  98.42217738,  98.43630768,
                98.45024185,  98.46413972,  98.4778477 ,  98.49153748,
                98.50510618,  98.51857347,  98.53187627,  98.54512187,
                98.55821708,  98.57121058,  98.58412028,  98.59687131,
                98.6095682 ,  98.62214947,  98.63454987,  98.64678565,
                98.6589859 ,  98.67117648,  98.68324896,  98.69521156,
                98.70701875,  98.71879355,  98.73047345,  98.74203536,
                98.75344601,  98.76472079,  98.77590417,  98.78703274,
                98.79812488,  98.80914888,  98.82010748,  98.83081679,
                98.84150442,  98.85205651,  98.86256317,  98.87300474,
                98.88337156,  98.89359853,  98.90379126,  98.91387187,
                98.92394505,  98.9338315 ,  98.94371491,  98.95346217,
                98.96316264,  98.97280588,  98.98239927,  98.99188594,
                99.0013173 ,  99.01068443,  99.0199978 ,  99.02930491,
                99.03849546,  99.04760031,  99.0566537 ,  99.06559931,
                99.07447689,  99.08329309,  99.09204521,  99.10077588,
                99.10942322,  99.11791105,  99.12637831,  99.13472485,
                99.14304107,  99.15131912,  99.15947975,  99.16761579,
                99.17571336,  99.1837393 ,  99.19170553,  99.19963292,
                99.20744427,  99.21522681,  99.22296711,  99.23066942,
                99.23828554,  99.24582993,  99.25334143,  99.2607881 ,
                99.26819624,  99.27556146,  99.28291144,  99.2901581 ,
                99.29732638,  99.30447173,  99.31154731,  99.31859652,
                99.32557812,  99.33249823,  99.33938697,  99.34620976,
                99.35301235,  99.35973698,  99.36635489,  99.37295094,
                99.37946828,  99.38597017,  99.39241583,  99.3988267 ,
                99.40518336,  99.41147395,  99.41773857,  99.4239522 ,
                99.43012148,  99.4362694 ,  99.44238385,  99.44847689,
                99.4544365 ,  99.46037862,  99.46626983,  99.47207949,
                99.4778837 ,  99.48364007,  99.48934982,  99.49503345,
                99.50064515,  99.50622716,  99.51175568,  99.51723333,
                99.52269087,  99.528124  ,  99.53348675,  99.53882917,
                99.54414562,  99.54941027,  99.55459632,  99.55977062,
                99.56493601,  99.57003887,  99.57512822,  99.58016876,
                99.585079  ,  99.5899511 ,  99.59479471,  99.59960306,
                99.60438877,  99.6091498 ,  99.61389599,  99.6185777 ,
                99.6231997 ,  99.62779071,  99.63235492,  99.6369083 ,
                99.6414162 ,  99.64589854,  99.65031904,  99.6547039 ,
                99.65906464,  99.66338037,  99.6676674 ,  99.67194965,
                99.67618793,  99.68040582,  99.68456449,  99.68871006,
                99.69283134,  99.69692462,  99.70098056,  99.70499253,
                99.70896912,  99.71289073,  99.71679487,  99.72068232,
                99.72453227,  99.7283155 ,  99.73209201,  99.73585697,
                99.73955554,  99.743233  ,  99.74687229,  99.7504692 ,
                99.75405575,  99.75761967,  99.7611598 ,  99.76467362,
                99.76817871,  99.77165906,  99.77509351,  99.77852148,
                99.78190668,  99.78526379,  99.7886066 ,  99.79191114,
                99.7951858 ,  99.79842998,  99.80166087,  99.80484423,
                99.80800273,  99.8111483 ,  99.81422472,  99.81728291,
                99.82032315,  99.82333948,  99.82633227,  99.82927887,
                99.83221812,  99.83512711,  99.83800651,  99.84087889,
                99.84373882,  99.84654962,  99.8493125 ,  99.85206791,
                99.85477974,  99.8574717 ,  99.86015008,  99.86278878,
                99.86541396,  99.8679983 ,  99.8705662 ,  99.87312481,
                99.87564887,  99.8781338 ,  99.88059805,  99.88304193,
                99.88547326,  99.88788661,  99.89029055,  99.89265077,
                99.89500588,  99.89734368,  99.89963464,  99.90191728,
                99.90418871,  99.90644468,  99.90866016,  99.9108612 ,
                99.91304955,  99.91519059,  99.9173216 ,  99.91943948,
                99.92152476,  99.92360601,  99.92565258,  99.92768446,
                99.92969349,  99.9316899 ,  99.93366352,  99.93561932,
                99.93754364,  99.93944212,  99.94132128,  99.94317424,
                99.94502218,  99.94684159,  99.94864469,  99.95042351,
                99.9521666 ,  99.95388993,  99.95559857,  99.95729477,
                99.9589592 ,  99.96060485,  99.96224141,  99.96386318,
                99.96547168,  99.96705543,  99.96861482,  99.97015474,
                99.9716634 ,  99.97314294,  99.97461149,  99.97606013,
                99.97749278,  99.9789158 ,  99.98031096,  99.98168949,
                99.98304613,  99.9843396 ,  99.98562388,  99.98690288,
                99.98813939,  99.98936182,  99.99054914,  99.99170908,
                99.9928478 ,  99.99397233,  99.99506115,  99.99612445,
                99.99716284,  99.99814597,  99.99909423, 100.        ])
```

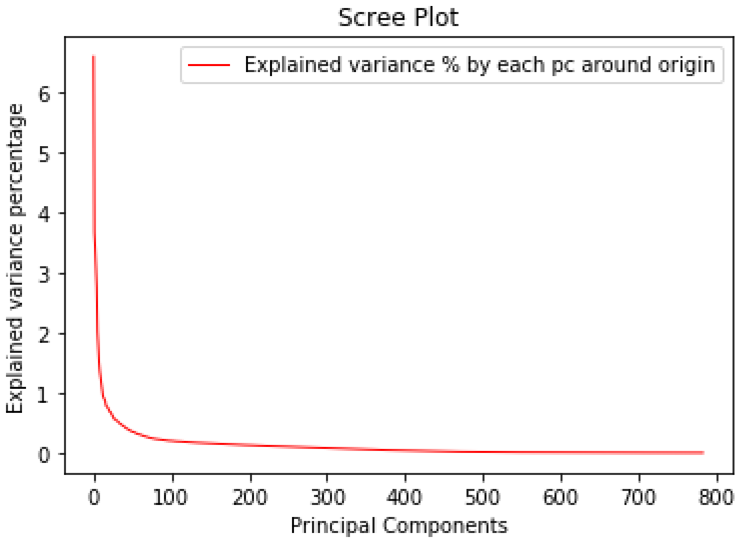
Explained variance % by each Principal component around origin.

In [28]:

```python
#sing_vals = np.arange(num_vars) + 1
plt.plot(explained_variance_proportion_variation_around_origin_for_each_pc*
100, 'r-', linewidth=1)
plt.title('Scree Plot')
plt.xlabel('Principal Components')
plt.ylabel('Explained variance percentage')
plt.legend(['Explained variance % by each pc around origin'], loc='best')
```

Out[28]:

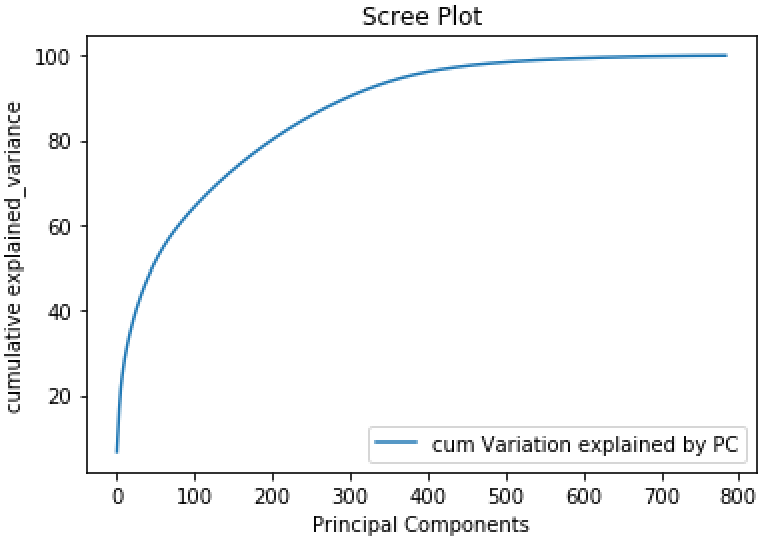
```
<matplotlib.legend.Legend at 0x1edbe8e6548>
```



Cumulative Explained variance % by each Principal component around origin.

```
In [29]: plt.plot(cum_exp_variance*100)
         plt.title('Scree Plot')
         plt.xlabel('Principal Components')
         plt.ylabel('cumulative explained_variance')
         plt.legend(['cum Variation explained by PC'], loc='best')
```

```
Out[29]: <matplotlib.legend.Legend at 0x1edbff1d888>
```

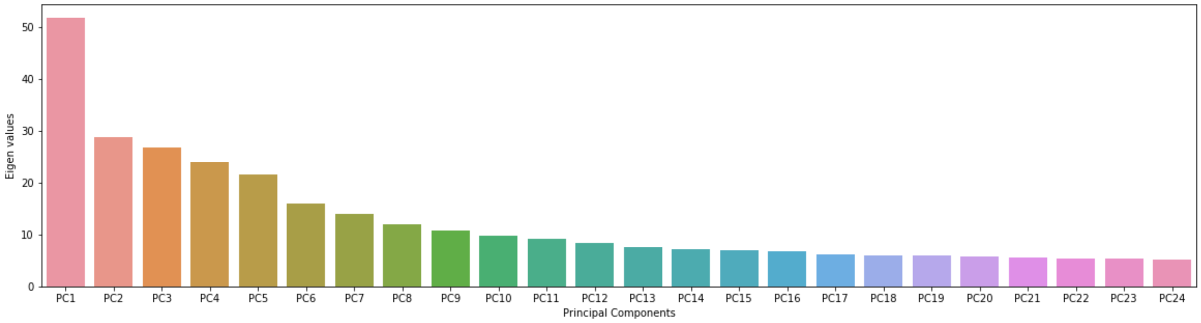


Eigen value/Amount of variance explained by each principal component.

In [30]:
```
plt.figure(figsize=(20,5))
val=values[0]
j=[]
for i in range(1,25):
    j.append('PC'+str(i))
p=pd.DataFrame(j,columns=['1'])
val=values[:25]
bar_df=pd.concat([val,p],axis=1)

sns.barplot(x='1',y=val[0],data=bar_df)
plt.xlabel('Principal Components')
plt.ylabel('Eigen values')
```

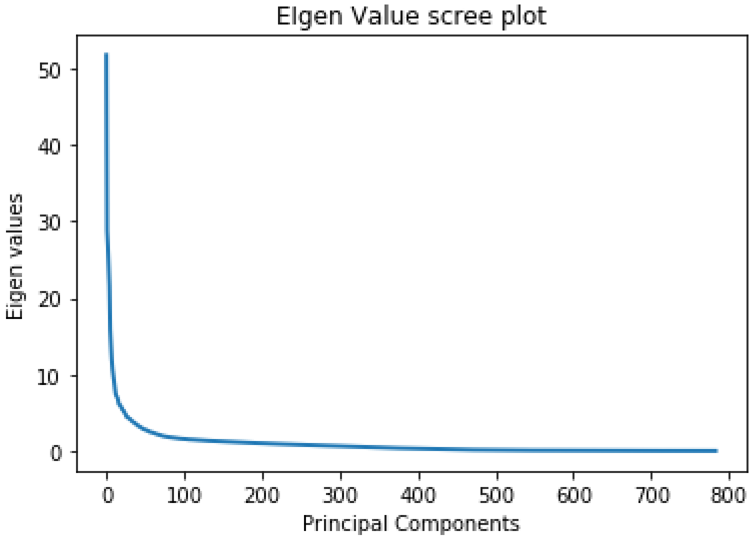
Out[30]:
```
Text(0, 0.5, 'Eigen values')
```



Eigen value scree plot

In [31]:
```
plt.plot(values, linewidth=2, markersize=12)
plt.xlabel('Principal Components')
plt.ylabel('Eigen values')
plt.title('EIgen Value scree plot')
```

Out[31]:
```
Text(0.5, 1.0, 'EIgen Value scree plot')
```



Checking lambda>1 for how many values.

In [32]:
```
values_more_than_1=values>1
```

In [33]:
```
values_more_than_1 = values_more_than_1.rename(columns={0: 'a'})
```

In [34]:
```
values_more_than_1.a.value_counts()
```

Out[34]:
```
False    585
True     199
Name: a, dtype: int64
```

Discussing Good cut-off

1. Scree Plots: Criteria-1

a)Eigen value scree plot-->

On visualizing the scree plot and using the elbow method to determine the number of principal components, we see that there is a big drop when PC=50(approximately). If we choose these number of components then the variance explained by our model is roughly around 40% which will be a drastic model. If we choose the point where the line becomes constant, then the number of components would be approximately 200 which suggests a good variance of around 80%.

b)Explained variance % scree plot-->

Same is the case with this plot as we are just finding the variance % explained by each P.C-->eigen.val/sum(eigen.val).

RESULT: we choose Number of components=200 (approximately)

2. Eigen value>1: Criteria-2

If we choose the principal components with eigen values i.e variance explained by each principal component to be more than 1, then we end up taking exactly 199 components which will let our model explain the variance of around 80%. shown in the above cell.

RESULT: we choose Number of components=199 or 200.

3. Cumulative explained variance: Criteria-3

If we want to have our model explain 80% of the total variance, then we simply analyze the Cumulative plot and find out that around 200 components will explain this much variance. Also, if we look the values of our variable "cum_exp_variance", it suggests 80% variance explained at No_components=200.

RESULT: we choose Number of components=200.

Citations:

1. https://towardsdatascience.com/let-us-understand-the-correlation-matrix-and-covariance-matrix-d42e6b643c22

2. https://ro-che.info/articles/2017-12-11-pca-explained-variance

3.https://stats.stackexchange.com/questions/22569/pca-and-proportion-of-variance-explained

```
In [ ]:
```

Part 2: subplots::=>

In [35]:

```
pca=PCA(20)
pca_transformed1=pca.fit_transform(df_features_znorm)
```

In [36]:

```
pca_transformed1
```

Out[36]:

```
array([[ 9.97069223, -6.18172062, -4.99286135, ..., -0.69373135,
        -1.1601097 ,  1.45042368],
       [11.41599975, -6.94158881, -5.06303443, ...,  0.17989547,
        -0.51211917,  0.16463441],
       [ 3.69011914, -4.69310042, -2.90865912, ...,  3.14749288,
         1.07171358,  4.42777009],
       ...,
       [-0.34942149, -0.9336773 ,  8.10744327, ..., -1.66920137,
        -1.20916765, -0.79560088],
       [-3.11526322, -2.09047048,  6.27252244, ..., -1.07402893,
        -0.33641313, -1.47238565],
       [-5.64409377,  0.24616446,  4.14018274, ...,  3.426435  ,
         0.59625527,  3.08214847]])
```

In [ ]:

```

```

In [37]:

```
final_data_components=pd.DataFrame(pca_transformed1)
```

In [38]:

```
# pca_transformed1.components_
```

In [39]:

```
final_data_components=pd.concat([final_data_components, df_full[['gnd']]],
axis = 1)
```

In [40]:
```
final_data_components
```

Out[40]:

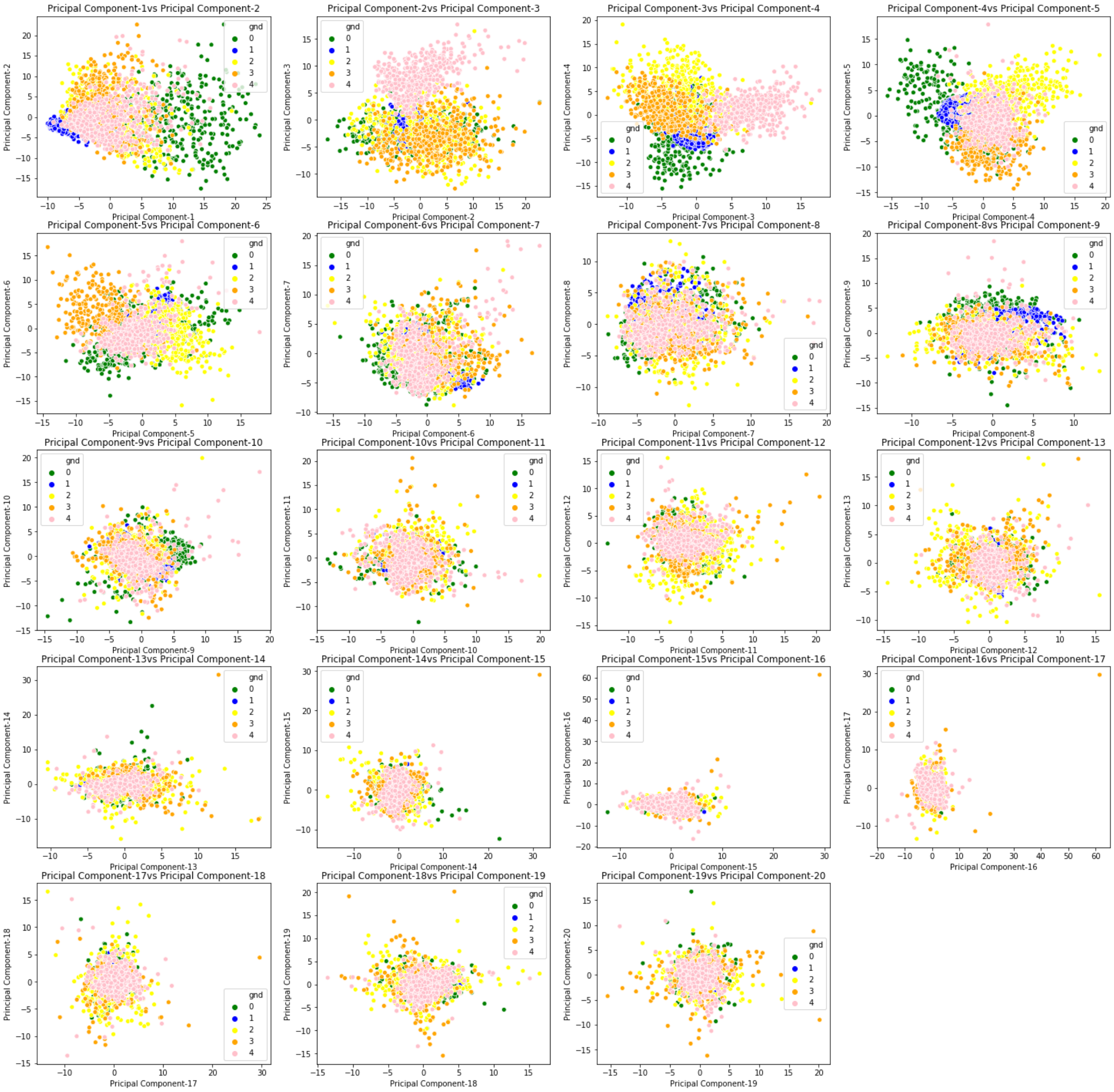
	0	1	2	3	4	5	6	7	
0	9.970692	-6.181721	-4.992861	-4.394894	-2.777859	-2.844865	1.276250	1.465378	6.8
1	11.416000	-6.941589	-5.063034	-4.242167	-1.844953	-0.146294	1.683077	2.973201	6.2
2	3.690119	-4.693100	-2.908659	3.935115	-6.811775	-3.226263	3.663473	5.407368	3.4
3	7.312408	-6.042890	-3.648012	3.506946	-4.562449	-5.523816	3.972068	4.890343	6.8
4	18.061520	-1.862430	-4.038829	-5.871645	-7.322167	4.442875	-2.186249	2.859818	2.5
...	...	...	...	...	...	...	...	...	
2061	-1.917926	-0.397486	7.951742	0.162117	-2.170054	-2.309860	-3.861855	-3.854849	1.3
2062	0.525752	2.646708	9.095311	0.969307	-2.257414	4.548021	-1.437043	-3.040765	-1.9
2063	-0.349421	-0.933677	8.107443	0.824251	-2.814456	-2.822887	-3.678484	-3.033047	-1.5
2064	-3.115263	-2.090470	6.272522	-1.529840	-0.036311	-2.586613	-2.099681	-0.086586	1.4
2065	-5.644094	0.246164	4.140183	2.238916	-2.477063	-3.198740	2.975437	3.480728	-2.8

2066 rows × 21 columns

In [41]:
```
fig, axs = plt.subplots(5,4,figsize=(25,25))
k=0
fig.delaxes(axs[4][3])
for i in range(0,axs.shape[0]):
    for j in range(0,axs.shape[1]):
        sns.scatterplot(final_data_components[k], final_data_components[k+1],hue=final_data_components.gnd,legend='full',ax=axs[i,j],palette=['green','blue','yellow','orange','pink'])
        axs[i,j].set(xlabel='Pricipal Component-'+str(k+1),ylabel='Principal Component-'+str(k+2))

        axs[i,j].set_title('Pricipal Component-'+str(k+1)+'vs Pricipal Component-'+str(k+2))

        if i==4 and j==2:
            break;
        k=k+1;
```



In [42]:
```
explained_variance_proportion_variation_around_origin_for_each_pc[0:15]
```

Out[42]:

	0
0	0.066011
1	0.036718
2	0.034130
3	0.030509
4	0.027506
5	0.020263
6	0.017672
7	0.015187
8	0.013592
9	0.012524
10	0.011619
11	0.010566
12	0.009524
13	0.009162
14	0.008947

In [43]:
```
corn1_g=final_data_components[[0,1,2,3,4,5,6,7,8,'gnd']].corr()
```

In [44]:
```
cor=corn1_g['gnd']
```

In [45]:
```
print("Correlation with class label::\n\n")
cor
```

```
Correlation with class label::



```

Out[45]:
```
0     -0.194708
1      0.156830
2      0.505208
3      0.398520
4     -0.251002
5      0.095011
6     -0.001216
7     -0.111017
8     -0.240323
gnd    1.000000
Name: gnd, dtype: float64
```

ANALYSIS: Dimensional cutoff

PC1 VS PC2::=>

1. Visually ::=>Variation between the data points is maximum. There is very less overlapping.

2. Amount of variance explained by each principal component is highest in case of pc1 and pc2 and the highest eigen values as well which is obvious.

3. Var_explained(PC1):6.6% , Var_explained(PC2):3.6% . Total explained variance by both the PC's: around 10%.

4. Although separation between the classes is not a parameter which PCA considers, but I have found out that correlation betweeen the class and the PC1 is 0.19, and the corr between PC2 and class is 0.15.

PC2 VS PC3::=>

1. Visually ::=>Variation between the data points is high with class 0 getting overlapped.

2. Amount of variance explained by each principal component in case of pc2 and pc3 is fair enough(Not too low).

3. Var_explained(PC2):3.36% , Var_explained(PC3):3.4% . Total explained variance by both the PC's: around 6.76%.

4. Although separation between the classes is not a parameter which PCA considers, but correlation betweeen the class and the PC2 is 0.09, and the corr between PC3 and class is 0.0012.

PC5 VS PC6::=>

1. Visually ::=>Variation between the data points is not too high with class 1 getting overlapped.

2. Amount of variance explained by each principal component in case of pc5 and pc6 is around 5%.

3. Var_explained(PC5):2.70% , Var_explained(PC6):2.0% . Total explained variance by both the PC's: around 4.70%.

4. Correlation betweeen the class and the PC5 is 0.25, and the corr between PC6 and class is 0.09 which is good enough and same as that provided by pc1 and pc2.

PC6 VS PC7::=>

1. Visually ::=>Variation between the data points is low with class 0 getting distored and spreaded.

2. Amount of variance explained by each principal component in case of pc6 and pc7 is around 3.7%.

3. Var_explained(PC6):2.0% , Var_explained(PC7):1.7% . Total explained variance by both the PC's: around 3.70%.

4. Correlation betweeen the class and the PC6 is 0.09, and the corr between PC7 and class is 0.0012 which is too low.

RESULT:==>

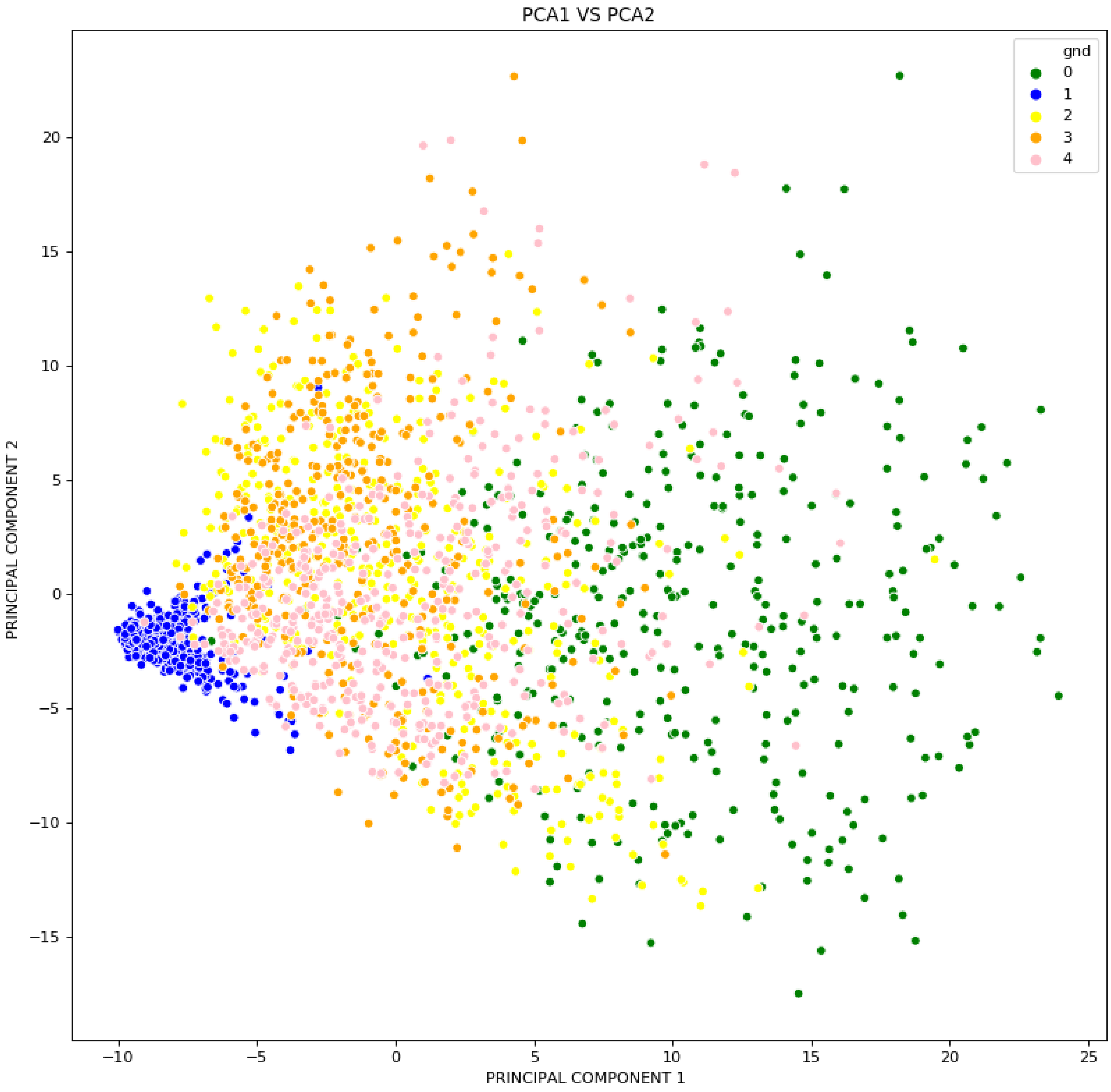
1.On comparing all the parameters (Visual cutoff, class separation, explained variance, eigen values,correlation between the component and the class), we have concluded that till PC5 and PC6, there is a good separability. After which explained variance and all the other parameters start to deteriorate as well as we can see from the plots. PC7 and PC8 overlaps the classes 0,2,3 with explained variance of only 3.2%

2. On comparing with the previous analysis, we find out that that if we want to achieve a variance of around 80% then we need to have 200 components approximately. And the total variance explained by the 6 components is around 21%. ALl the other components after this, comtribute very little to the explained variance which can be seen by our variable "cum_exp_variance".

3. PCA 1 vs PCA 2 AND PCA 5 vs PCA 6 => Explain the results versus the known classes and compare between the two plots.

In [46]:
```
fig2=plt.figure(figsize=(12,12), dpi= 80, facecolor='w', edgecolor='k')
sns.scatterplot(final_data_components[0],final_data_components[1],hue=final_data_components.gnd,legend='full',palette=['green','blue','yellow','orange','pink'])
plt.xlabel("PRINCIPAL COMPONENT 1 ")
plt.ylabel("PRINCIPAL COMPONENT 2 ")
plt.title("PCA1 VS PCA2")
```

Out[46]:
```
Text(0.5, 1.0, 'PCA1 VS PCA2')
```

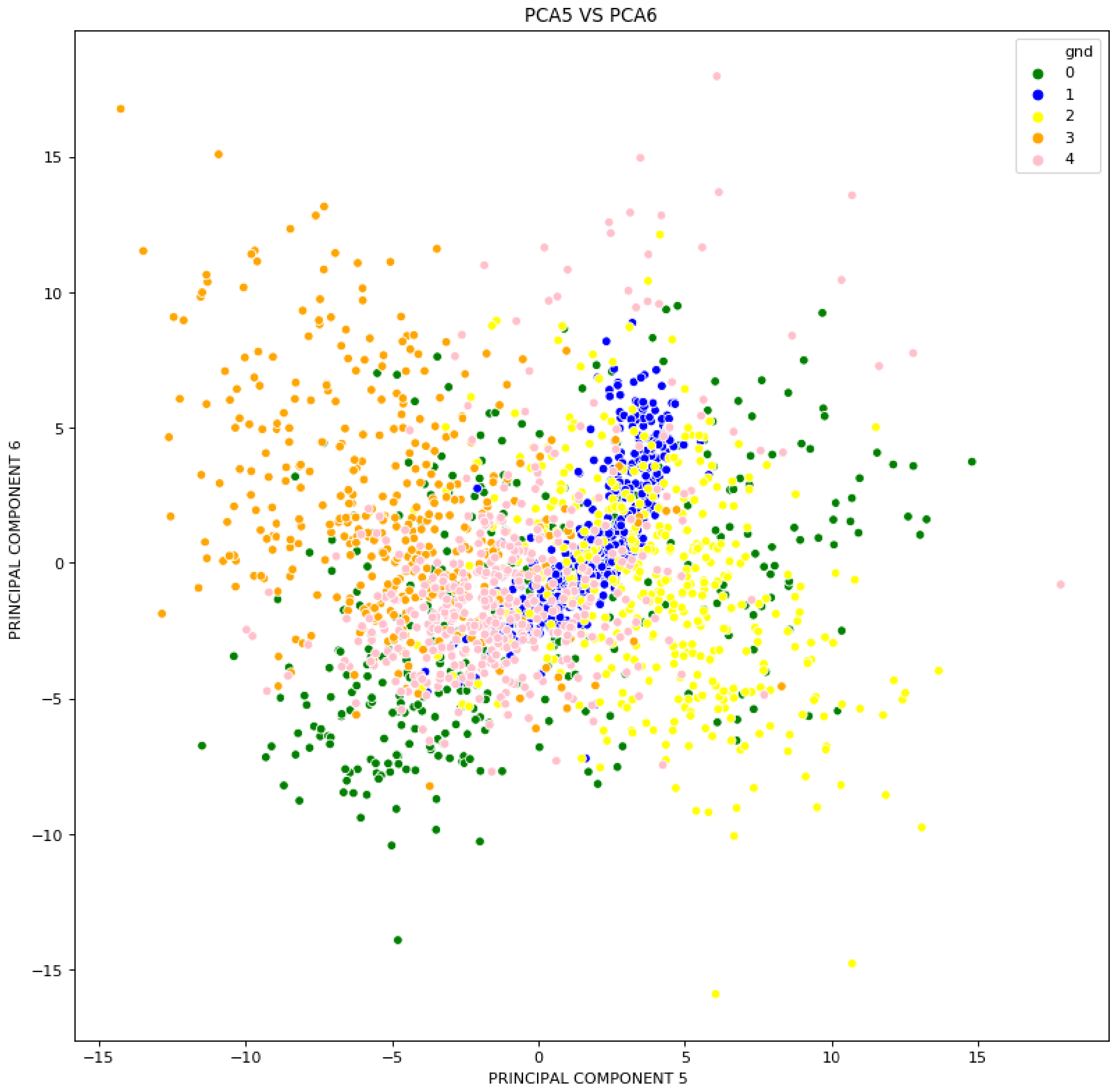


In [47]:

```
#2.2.1.3
fig3=plt.figure(figsize=(12,12), dpi= 80, facecolor='w', edgecolor='k')
sns.scatterplot(final_data_components[4],final_data_components[5],hue=final
_data_components.gnd,legend='full',palette=['green','blue','yellow','orang
e','pink'])
plt.xlabel("PRINCIPAL COMPONENT 5 ")
plt.ylabel("PRINCIPAL COMPONENT 6 ")
plt.title("PCA5 VS PCA6")
```

Out[47]:

```
Text(0.5, 1.0, 'PCA5 VS PCA6')
```



ANALYSIS ::=>

PC1 VS PC2:: RESULTS VERSUS KNOWN CLASSES=>

1. Visually ::=>

1.class 0: There is no cluster being formed for class 0 and there is no overlapping as well.

2.class 1: It is clearly custered and well separated.

3.class [2,3,4]: These classes are getting overlapped with each other.

RESULT :=>The amount of variance explained by each PC is maximum as compared to other components(Var_explained(PC1):6.6% ,Var_explained(PC2):3.6%) because these two combined covers the maximum explained variance. As PCA doesn't take class labels into account, so the overlapping of the classes is compromised in the plots.(correlation betweeen the class and the PC1: 0.19, and the correlation between PC2 and class: 0.15.)

PC5 VS PC6:: RESULTS VERSUS KNOWN CLASSES=>

1. Visually ::=>

1.class 0: This is distorted and heavily spread.

2.class 1: This class is overlapped by [2,4].

3.class [3]: This is somehow separated.

RESULT :=>The amount of variance explained by each PC (Var_explained(PC5):2.70% , Var_explained(PC6):2.0%) which is comparatively bad than PC1 and PC2.Correlation betweeen the class and the PC5: 0.25, and the correlation between PC6 and class: 0.09.)

COMPARISON BETWEEN THE TWO PLOTS:==>

1.Class 1 is getting clearly separated in [PC1,PC2] plot, whereas class 1 is getting overlapped in [PC5,PC6].

2.In [PC1,PC2]-> class 0 is also separated with little bit of spread whereas in [PC5,PC6], it is distoreted and overlapped.

3.Overall correlation between [PC1,class]+[PC2,class] is same as [PC5,class]+[PC6,class] which is 0.34.

4.Overall explained_variance[PC1,PC2]={10.2%} and explained_variance[PC5,PC6]={4.7%} which shows data points are well varied in PC1,PC2 plot which can be seen in the plots because the variance explained by each of them is highest.

4) PCA AND DUAL PCA

In [48]:
```
import timeit
start = timeit.default_timer()
```

In [49]:
```
#finding cov matrix for features. i.e d*d
# covariance_matrix_manual=np.cov(df_features_znorm.T) # transpose of datas
et i.e features on rows side. equivalent to AT.A i.e on doing eig , vector
 it gives is right singular vector feature ke along
# covariance_matrix_manual1=pd.DataFrame(covariance_matrix_manual)
# df_features_znorm_transpose=np.transpose(df_features_znorm)
df_features_znorm_transpose=df_features_znorm.T
df_features_znorm_np=df_features_znorm.to_numpy()
df_features_znorm_np_T=df_features_znorm_transpose.to_numpy()
```

In [50]:
```
#finding cov matrix at*a --> gives us U i.e features*features 784*784
cov_mat1=np.matmul(df_features_znorm_np_T,df_features_znorm_np)
# cov_mat1=cov_mat1/(df_full.shape[0]-1) # also not feasible for pca projec
tion dont do BECAUSE OF COMPARISON WITH S GIVEN BY NUMPY SVD (IT DOESNT DIV
IDE VALUES BY N-1).
values_pca1, vectors_pca1= np.linalg.eig(cov_mat1)
#vectors_pca1 #U unsorted 784*784
```

In [51]:
```
vectors_pca1
```

Out[51]:
```
array([[-0.00197863,  0.00493308, -0.00037529, ..., -0.00013358,
        -0.00062295, -0.00157864],
       [-0.00151307, -0.00640373,  0.00258725, ...,  0.00274212,
        -0.00837347,  0.0063992 ],
       [ 0.00049178, -0.00156563, -0.00372451, ...,  0.00445827,
         0.00683485, -0.00181313],
       ...,
       [ 0.0001125 ,  0.00300533, -0.00335936, ...,  0.00245866,
        -0.00039547,  0.00083265],
       [ 0.00132315,  0.00947149,  0.00553066, ..., -0.01026383,
        -0.01070597,  0.01537859],
       [-0.00591181,  0.00287621,  0.00624184, ...,  0.00383851,
        -0.00246361,  0.00327846]])
```

In [52]:
```
# #finding cov matrix a*at  --> gives us U when passed into eigh i.e observ
ations*observations 2066*2066
# cov_mat=np.matmul(df_features_znorm_np,df_features_znorm_np_T)
# cov_mat=cov_mat/(df_full.shape[0]-1)
# values_pca, vectors_pca = np.linalg.eigh(cov_mat)
# #vectors_pca #u sorted 2066*2066
# cov_mat=pd.DataFrame(cov_mat)
# # cov_mat
# sorted_values2 = np.argsort(values_pca)[::-1]
# vectors_pca = vectors_pca[:,sorted_values2]
# values_pca = values_pca[sorted_values2]
# values_pca=pd.DataFrame(values_pca)
# sorted_vectors2=pd.DataFrame(vectors_pca.T)
# v=sorted_vectors2
# v # equivalent to uh in svd
```

In [53]:
```
cov_mat1=pd.DataFrame(cov_mat1)
cov_mat1
```

Out[53]:

	0	1	2	3	4	5	6
0	2066.000000	-41.823743	57.874052	-57.199348	-6.523476	55.229393	-14.783244
1	-41.823743	2066.000000	44.760228	-68.776377	-17.131647	97.521100	3.914803
2	57.874052	44.760228	2066.000000	0.991612	-46.487979	-19.050772	-34.701012
3	-57.199348	-68.776377	0.991612	2066.000000	16.739481	-20.104114	-62.806372
4	-6.523476	-17.131647	-46.487979	16.739481	2066.000000	-97.356457	-104.750265
...	...	...	...	...	...	...	...
779	0.467566	41.243121	98.016048	15.180252	20.994929	-30.593971	17.570093
780	-25.518658	-27.541538	41.750798	-8.986629	68.180045	-64.300539	-21.734513
781	1.264345	-20.144299	12.484072	-14.431025	-12.729503	-11.266006	40.744245
782	27.349291	74.631377	-57.230211	11.942206	14.714233	-0.216029	41.312080
783	65.592079	-37.354962	64.114544	-6.604229	3.782801	-14.702626	17.741478

784 rows × 784 columns

-sorting the vectors and values -

In [54]:
```
# sorted_values3 = np.argsort(values_pca1)[::-1]
# vectors_pca1 = vectors_pca1[:,sorted_values3]
# values_pca1 = values_pca1[sorted_values3]
# values_pca1=pd.DataFrame(values_pca1)
# sorted_vectors3=pd.DataFrame(vectors_pca1.T)
# u=sorted_vectors3
```

In [55]:
```
u=vectors_pca1 #equivalent to vh in svd
```

In [56]:
```
u
```

Out[56]:
```
array([[-0.00197863,  0.00493308, -0.00037529, ..., -0.00013358,
        -0.00062295, -0.00157864],
       [-0.00151307, -0.00640373,  0.00258725, ...,  0.00274212,
        -0.00837347,  0.0063992 ],
       [ 0.00049178, -0.00156563, -0.00372451, ...,  0.00445827,
         0.00683485, -0.00181313],
       ...,
       [ 0.0001125 ,  0.00300533, -0.00335936, ...,  0.00245866,
        -0.00039547,  0.00083265],
       [ 0.00132315,  0.00947149,  0.00553066, ..., -0.01026383,
        -0.01070597,  0.01537859],
       [-0.00591181,  0.00287621,  0.00624184, ...,  0.00383851,
        -0.00246361,  0.00327846]])
```

In [57]:
```
u.shape
```

Out[57]:
```
(784, 784)
```

In [58]:
```
df_features_znorm.shape
```

Out[58]:
```
(2066, 784)
```

In [59]:
```
s=np.sqrt(values_pca1)
```

In [60]:
```
s=pd.DataFrame(s)
```

In [61]:
```
s
```

Out[61]:

	0
0	326.986490
1	243.872478
2	235.121097
3	222.297469
4	211.074528
...	...
779	7.451470
780	6.586566
781	7.404827
782	7.017413
783	6.899872

784 rows × 1 columns

In [62]:
```
df_features_znorm
```

Out[62]:

	fea.1	fea.2	fea.3	fea.4	fea.5	fea.6	fea.7	fea.8	
0	1.010077	0.966782	0.359594	-1.668004	-1.638671	1.007994	-0.324102	-0.992537	0.9
1	1.687176	-1.029924	1.026488	0.336317	-0.976018	0.340307	1.674690	-0.992537	0.9
2	-1.021220	0.301213	-1.641090	0.336317	-0.976018	-0.995067	-1.656630	-0.992537	-1.6
3	1.687176	0.301213	-0.307301	0.336317	1.674594	-0.327380	-0.324102	-1.648724	0.9
4	0.332978	1.632350	0.359594	0.336317	-1.638671	1.007994	-0.990366	-0.992537	0.9
...	...	...	...	...	...	...	...	...	
2061	1.010077	-1.695492	0.359594	-1.668004	1.011941	-1.662753	1.008426	0.319835	-1.0
2062	-0.344121	-0.364355	0.359594	1.004424	-0.313365	-0.995067	-0.324102	0.319835	0.3
2063	-0.344121	0.301213	-0.307301	0.336317	-0.976018	-0.327380	1.674690	1.632208	1.6
2064	1.687176	-0.364355	1.026488	0.336317	-0.976018	-1.662753	0.342162	-0.336351	-0.3
2065	0.332978	0.301213	-0.974195	0.336317	-0.313365	1.675681	1.008426	-0.336351	-0.3

2066 rows × 784 columns

In [63]:
```
proj_pca_=np.dot(u.T,df_features_znorm_np_T)
```

In [64]:
```
proj_pca=pd.DataFrame(proj_pca_)
```

Projected data is found using utxt or ut.x

In [65]:
```
proj_pca.T
```

Out[65]:

```
              0          1          2          3          4          5          6          7
   0   -9.970692   6.181722  -4.992863   4.394879   2.777870  -2.844761   1.276232   1.463996   6.8
   1  -11.416000   6.941587  -5.063029   4.242178   1.844940  -0.146431   1.683401   2.974999   6.2
   2   -3.690119   4.693097  -2.908656  -3.935094   6.811806  -3.226107   3.662990   5.408198   3.4
   3   -7.312408   6.042886  -3.648012  -3.506919   4.562468  -5.523882   3.971637   4.891807   6.7
   4  -18.061520   1.862436  -4.038834   5.871621   7.322140   4.443076  -2.186078   2.857345   2.5
 ...         ...        ...        ...        ...        ...        ...        ...        ...
2061    1.917926   0.397488   7.951743  -0.162132   2.170046  -2.309800  -3.861676  -3.855531   1.3
2062   -0.525752  -2.646714   9.095319  -0.969272   2.257428   4.547903  -1.437164  -3.038324  -1.9
2063    0.349422   0.933681   8.107442  -0.824268   2.814438  -2.822867  -3.678327  -3.033837  -1.5
2064    3.115263   2.090474   6.272519   1.529823   0.036308  -2.586437  -2.099663  -0.088302   1.4
2065    5.644094  -0.246167   4.140183  -2.238905   2.477097  -3.198489   2.975116   3.481201  -2.8

2066 rows × 784 columns
```

Reconstruction ut.yt or u.y

In [66]:
```
X_hat=np.dot(u,proj_pca)
```

In [67]:
```
X_hat=pd.DataFrame(X_hat)
```

In [68]:
```
X_hat.T
```

Out[68]:

	0	1	2	3	4	5	6	7	
0	1.010077	0.966782	0.359594	-1.668004	-1.638671	1.007994	-0.324102	-0.992537	0.9
1	1.687176	-1.029924	1.026488	0.336317	-0.976018	0.340307	1.674690	-0.992537	0.9
2	-1.021220	0.301213	-1.641090	0.336317	-0.976018	-0.995067	-1.656630	-0.992537	-1.6
3	1.687176	0.301213	-0.307301	0.336317	1.674594	-0.327380	-0.324102	-1.648724	0.9
4	0.332978	1.632350	0.359594	0.336317	-1.638671	1.007994	-0.990366	-0.992537	0.9
...	...	...	...	...	...	...	...	...	
2061	1.010077	-1.695492	0.359594	-1.668004	1.011941	-1.662753	1.008426	0.319835	-1.0
2062	-0.344121	-0.364355	0.359594	1.004424	-0.313365	-0.995067	-0.324102	0.319835	0.3
2063	-0.344121	0.301213	-0.307301	0.336317	-0.976018	-0.327380	1.674690	1.632208	1.6
2064	1.687176	-0.364355	1.026488	0.336317	-0.976018	-1.662753	0.342162	-0.336351	-0.3
2065	0.332978	0.301213	-0.974195	0.336317	-0.313365	1.675681	1.008426	-0.336351	-0.3

2066 rows × 784 columns

In [69]:
```
df_features_znorm
```

Out[69]:

	fea.1	fea.2	fea.3	fea.4	fea.5	fea.6	fea.7	fea.8	
0	1.010077	0.966782	0.359594	-1.668004	-1.638671	1.007994	-0.324102	-0.992537	0.9
1	1.687176	-1.029924	1.026488	0.336317	-0.976018	0.340307	1.674690	-0.992537	0.9
2	-1.021220	0.301213	-1.641090	0.336317	-0.976018	-0.995067	-1.656630	-0.992537	-1.6
3	1.687176	0.301213	-0.307301	0.336317	1.674594	-0.327380	-0.324102	-1.648724	0.9
4	0.332978	1.632350	0.359594	0.336317	-1.638671	1.007994	-0.990366	-0.992537	0.9
...	...	...	...	...	...	...	...	...	
2061	1.010077	-1.695492	0.359594	-1.668004	1.011941	-1.662753	1.008426	0.319835	-1.0
2062	-0.344121	-0.364355	0.359594	1.004424	-0.313365	-0.995067	-0.324102	0.319835	0.3
2063	-0.344121	0.301213	-0.307301	0.336317	-0.976018	-0.327380	1.674690	1.632208	1.6
2064	1.687176	-0.364355	1.026488	0.336317	-0.976018	-1.662753	0.342162	-0.336351	-0.3
2065	0.332978	0.301213	-0.974195	0.336317	-0.313365	1.675681	1.008426	-0.336351	-0.3

2066 rows × 784 columns

RESULT ::=>

1.Reconstruction is done.

2.The original dataset is equal to x_hat. (Reconstructed)

3.Singular value s and singular vector u is also found which can be compared with implementation of

In [70]:

```
stop = timeit.default_timer()
print('MODEL: ', stop - start)

Timetaken = pd.DataFrame(columns = ['Model', 'time'])
Timetaken = Timetaken.append({'Model':'PCA', 'time':stop-start},ignore_inde
x=True)
```

```
MODEL:  3.7157768110000013
```

In [71]:

```
Timetaken
```

Out[71]:

	Model	time
0	PCA	3.715777

In [72]:

```
print('SINGUALAR VALUES')
print(s)
```

```
SINGUALAR VALUES
              0
0    326.986490
1    243.872478
2    235.121097
3    222.297469
4    211.074528
..          ...
779    7.451470
780    6.586566
781    7.404827
782    7.017413
783    6.899872

[784 rows x 1 columns]
```

Using SVD to validate the above results only

In [73]:

```
import scipy
```

In [74]:

```
u, Sigma,v = np.linalg.svd(df_features_znorm,full_matrices=True)
```

In [75]:
```
Uh=pd.DataFrame(u)
Vh=pd.DataFrame(v)
Sigma=pd.DataFrame(Sigma)
```

In [76]:
```
Uh #equivalent to v in dual pca
```

Out[76]:

	0	1	2	3	4	5	6	7	
0	0.030493	-0.025348	0.021235	0.019770	0.013161	-0.015703	0.007543	0.009334	-0.04
1	0.034913	-0.028464	0.021534	0.019083	0.008741	-0.000808	0.009950	0.018968	-0.04
2	0.011285	-0.019244	0.012371	-0.017702	0.032272	-0.017808	0.021650	0.034482	-0.0
3	0.022363	-0.024779	0.015515	-0.015776	0.021615	-0.030491	0.023475	0.031190	-0.04
4	0.055236	-0.007637	0.017178	0.026413	0.034690	0.024525	-0.012921	0.018218	-0.0
...	...	...	...	...	...	...	...	...	
2061	-0.005865	-0.001630	-0.033820	-0.000729	0.010281	-0.012750	-0.022825	-0.024583	-0.0
2062	0.001608	0.010853	-0.038684	-0.004360	0.010695	0.025104	-0.008494	-0.019372	0.0
2063	-0.001069	-0.003829	-0.034482	-0.003708	0.013334	-0.015582	-0.021741	-0.019344	0.0
2064	-0.009527	-0.008572	-0.026678	0.006882	0.000172	-0.014277	-0.012410	-0.000563	-0.0
2065	-0.017261	0.001009	-0.017609	-0.010072	0.011736	-0.017655	0.017585	0.022196	0.0

2066 rows × 2066 columns

In [77]:
```
Vh #equivalent to u in pca
```

Out[77]:

	0	1	2	3	4	5	6	7	
0	0.001979	0.001513	-0.000492	-0.003617	-0.000810	0.005896	-0.001439	-0.001285	-0.00
1	-0.004933	0.006404	0.001566	-0.000518	0.002937	0.003685	0.008790	-0.000596	0.00
2	0.000375	-0.002587	0.003725	-0.000352	0.001220	0.000092	0.002634	-0.005623	0.00
3	-0.006154	-0.002790	0.003226	0.002489	-0.010679	-0.000604	0.001788	0.003557	0.00
4	0.003251	-0.003767	-0.001384	0.000363	0.001331	-0.008793	-0.002887	-0.007900	0.00
...	...	...	...	...	...	...	...	...	
779	-0.000990	-0.002825	-0.002969	-0.007797	-0.005491	-0.001475	-0.001826	-0.004143	0.00
780	0.003076	0.002147	-0.000022	-0.002503	-0.006257	0.003685	-0.000386	-0.003653	0.00
781	-0.002942	-0.001887	-0.008383	0.001179	0.004695	0.002205	0.002482	0.000407	0.00
782	0.000692	-0.004045	-0.000907	-0.004380	-0.003882	0.000384	0.000257	-0.000715	0.00
783	-0.001785	-0.004795	0.008164	-0.004050	0.004779	0.001970	0.007182	-0.003580	-0.00

784 rows × 784 columns

In [78]:
```
Sigma
```

Out[78]:
```
              0
  0  326.986490
  1  243.872478
  2  235.121097
  3  222.297469
  4  211.074528
...         ...
779    4.150039
780    4.101121
781    3.990520
782    3.919102
783    3.830292

784 rows × 1 columns
```

Dual PCA

In [79]:
```
import timeit
start = timeit.default_timer()
```

In [80]:
```
#finding cov matrix a*at  --> gives us U when passed into eigh i.e observat
ions*observations 2066*2066
cov_mat=np.matmul(df_features_znorm_np,df_features_znorm_np_T)
# cov_mat=cov_mat/(df_full.shape[0]-1)
values_pca, vectors_pca = np.linalg.eigh(cov_mat)
#vectors_pca #u sorted 2066*2066
```

In [81]:
```
cov_mat=pd.DataFrame(cov_mat)
```

In [82]:
```
sorted_values2 = np.argsort(values_pca)[::-1]
vectors_pca = vectors_pca[:,sorted_values2]
values_pca = values_pca[sorted_values2]
values_pca=pd.DataFrame(values_pca)
sorted_vectors2=pd.DataFrame(vectors_pca.T)
v=sorted_vectors2
```

In [83]:
```
v # equivalent to uh in svd
```

Out[83]:

	0	1	2	3	4	5	6	7	
0	0.030493	0.034913	0.011285	0.022363	0.055236	0.046916	0.040937	0.021700	0.0
1	-0.025348	-0.028464	-0.019244	-0.024779	-0.007637	0.032482	-0.021805	0.042834	0.0
2	0.021235	0.021534	0.012371	0.015515	0.017178	0.004994	0.024493	0.007139	0.0
3	-0.019770	-0.019083	0.017702	0.015776	-0.026413	-0.033458	-0.011377	-0.031623	0.0
4	0.013161	0.008741	0.032272	0.021615	0.034690	-0.004860	0.039286	-0.054772	0.0
...	...	...	...	...	...	...	...	...	
2061	0.000000	0.020181	-0.006487	-0.024432	-0.044182	-0.084449	0.385945	-0.066391	-0.3
2062	0.000000	0.002232	0.038900	-0.148384	0.137218	0.173324	0.038450	-0.248543	0.1
2063	0.000000	0.012837	0.032139	-0.157285	0.087955	-0.082976	-0.152485	0.250372	-0.0
2064	0.000000	0.007584	-0.049012	-0.124722	-0.103469	0.143160	-0.106512	0.177942	-0.1
2065	0.000000	-0.124867	-0.019983	-0.013534	-0.170539	-0.397274	-0.052720	-0.184384	0.1

2066 rows × 2066 columns

In [84]:
```
s1=np.sqrt(values_pca[:784])
values_pca=values_pca.to_numpy()
```

In [85]:
```
s1
```

Out[85]:

	0
0	326.986490
1	243.872478
2	235.121097
3	222.297469
4	211.074528
...	...
779	4.150039
780	4.101121
781	3.990520
782	3.919102
783	3.830292

784 rows × 1 columns

In [86]:
```
values_pca_list=values_pca.tolist()
```

In [87]:

```
values_pca.shape
```

Out[87]:

```
(2066, 1)
```

In [88]:

```
flat_list = []
for sublist in values_pca_list:
    for item in sublist:
        flat_list.append(item)
```

In [89]:

```
flat_list
```

```
Out[89]: [106920.16459705515,
          59473.78546651998,
          55281.93027001767,
          49416.16484289528,
          44552.456335586045,
          32820.12934635171,
          28624.908751809486,
          24598.463283754376,
          22015.82883906395,
          20285.698577942545,
          18819.039266814736,
          17114.634137768484,
          15426.005914921076,
          14840.859475957686,
          14492.618776370831,
          13895.87853319045,
          12744.588611643063,
          12452.682943678847,
          12332.568618996762,
          11865.252721872037,
          11494.656874874578,
          11028.866103400273,
          10965.144799081558,
          10474.59816615751,
          10123.042920923264,
          9788.776585319629,
          9210.745699190167,
          9166.035977787553,
          9034.849403819424,
          8776.074334044573,
          8667.551374830231,
          8532.39818663617,
          8273.801850424565,
          8052.670661098706,
          7890.442375732005,
          7760.306615935903,
          7642.301127614923,
          7377.061645909894,
          7307.43054299972,
          7178.131797530683,
          6976.813060864306,
          6767.29018266696,
          6600.05224166115,
          6582.43694369389,
          6401.141612382335,
          6208.321960844764,
          6105.551174688178,
          6063.1133565912805,
          5878.382178508136,
          5792.127585266645,
          5735.587720907892,
          5649.142794341403,
          5638.732731231177,
          5452.2952913278,
          5373.9956724690965,
          5297.890012858874,
          5127.124387147885,
 5091.186916041768,
 5035.793325319473,
 4977.610293332051,
 4927.145136230499,
 4841.05230506824,
 4772.137892662617,
 4664.136708559146,
 4604.9854132920855,
 4540.860932212165,
 4447.6821482903315,
 4430.942574831066,
 4322.521730887325,
 4260.49428145748,
 4238.516820834206,
 4142.087069427805,
 4067.235987498035,
 4040.36925254576,
 3977.3404782066423,
 3932.0338067223997,
 3883.591579852994,
 3806.4069504402923,
 3784.3580655787973,
 3750.877550308347,
 3735.4795548887187,
 3717.779919812464,
 3699.312346022411,
 3672.020634651196,
 3620.8553128343933,
 3601.7860651243473,
 3577.3130599593724,
 3545.6715237207795,
 3501.433776069864,
 3462.184453668407,
 3452.783151445109,
 3436.442963034464,
 3413.4133899589433,
 3399.151268424607,
 3358.8919351335257,
 3325.135267972784,
 3298.3176233682198,
 3290.03557059407,
 3266.175267389505,
 3238.419644272103,
 3217.138837413422,
 3214.157346297226,
 3203.367288130663,
 3169.331930832375,
 3143.9799964730005,
 3135.582301693815,
 3108.376212197097,
 3105.823716009211,
 3094.04891377295,
 3078.58779585424,
 3058.4223243422553,
 3049.849289061528,
 3038.524618493654,
 3028.2192888696036,
 2993.5312581286657,
 2984.0228179731794,
 2973.6355509480013,
 2946.8606013313333,
 2937.5342308596337,
 2924.35497166885,
 2908.1857689279955,
 2895.628168890785,
 2882.1231302266724,
 2867.865464960211,
 2847.7337948060967,
 2845.375977793418,
 2820.384134805139,
 2816.0832933246684,
 2800.315035099274,
 2783.4507696574074,
 2764.8203671968813,
 2746.692869785681,
 2742.39652961327,
 2731.3109711910224,
 2724.2710052697794,
 2698.20012013966,
 2692.3092685730776,
 2682.453575603247,
 2678.318969752775,
 2668.6069510111515,
 2649.754320987274,
 2640.506532549421,
 2628.9185469883587,
 2613.381692549348,
 2597.877493563606,
 2590.0912809928836,
 2582.665172509923,
 2571.9500252521507,
 2562.3129515191163,
 2541.5260291294253,
 2537.9308336652675,
 2524.3863518279777,
 2512.9045797129893,
 2499.118990283301,
 2487.0724120559494,
 2471.6857664855133,
 2470.0501098688956,
 2456.470128179988,
 2446.051723995311,
 2435.1674658358384,
 2429.3281404673535,
 2409.870653744171,
 2396.9592584382804,
 2392.006691176002,
 2382.9287893433802,
 2378.672859093568,
 2362.0352570110977,
 2355.528710851534,
 2348.9857719337447,
 2337.3650672989847,
 2336.339530526359,
                2321.4872922417826,
                2309.244269521199,
                2303.4820970041314,
                2294.1976104129008,
                2274.369358307413,
                2265.1175882158286,
                2250.691823281561,
                2241.393558500533,
                2238.524013547734,
                2221.38674022203,
                2214.4842878990758,
                2210.4633607623005,
                2196.0966352933688,
                2187.1674073047066,
                2177.805895355231,
                2173.7078332129495,
                2166.2625454161134,
                2158.9944486497498,
                2152.645545557593,
                2138.3433398784914,
                2128.174283673508,
                2122.6833472918397,
                2118.779087084499,
                2107.5950563540064,
                2104.7894278632425,
                2093.9011433809196,
                2077.548849840784,
                2065.721353768416,
                2059.3027979571743,
                2057.359863176664,
                2049.518400203888,
                2042.5908352565475,
                2039.5457668414688,
                2034.7806462998988,
                2028.0111449256035,
                2010.356505155948,
                2004.0509882283075,
                2000.058652327822,
                1986.651725912639,
                1980.9101089655517,
                1971.541785845444,
                1959.4861218462675,
                1955.8648068330301,
                1950.9419062470924,
                1944.3067828382977,
                1933.2925662841951,
                1927.734517397361,
                1914.5304724614916,
                1898.0176125999633,
                1890.1296784072108,
                1878.4871388305494,
                1871.8983233221954,
                1867.4160983760794,
                1860.5063392005368,
                1850.1376547227867,
                1844.960365967413,
                1832.016189667832,
                1829.9040735126916,
                1820.5057203862295,
                1808.0117122698466,
                1802.5989969670477,
                1792.6393365075787,
                1789.2966658644493,
                1781.6092507231551,
                1769.2543871976513,
                1767.983703653112,
                1763.1668810196509,
                1753.46205362002,
                1748.2942723751164,
                1734.8211404105323,
                1718.1240877659345,
                1713.5318606623564,
                1712.3530223796447,
                1703.4365227670573,
                1698.7574592181445,
                1693.1747660664694,
                1685.4393471288774,
                1669.5252732705183,
                1667.099093024676,
                1665.051743697216,
                1647.444666371117,
                1646.915963996808,
                1630.6484139648085,
                1627.7527039573797,
                1622.158299575194,
                1618.2390289946413,
                1609.2602621042884,
                1598.353408433035,
                1583.5176090418215,
                1580.1526602103306,
                1571.879464328683,
                1560.8935861534826,
                1554.6785183651969,
                1550.3854875298164,
                1547.2251527092512,
                1535.7399913253387,
                1528.220147048032,
                1520.5989656747768,
                1515.9791405397482,
                1506.942245939231,
                1500.5726292096913,
                1498.2684937583867,
                1489.5154121138826,
                1475.4101738964628,
                1474.9223413207017,
                1465.3334083204747,
                1459.4802829752075,
                1453.1700810808677,
                1450.5617441241855,
                1439.956753992306,
                1428.5158381174303,
                1425.898932706145,
                1418.9079927245293,
                1410.8165476946253,
                    1390.432132838342,
                    1387.8123430401893,
                    1385.2762154394923,
                    1378.293179971012,
                    1375.0866201645854,
                    1370.2382551363169,
                    1361.3125416691987,
                    1355.6428596192386,
                    1350.002658274157,
                    1341.5511008749072,
                    1332.992581129855,
                    1324.9546545657067,
                    1319.6924906129645,
                    1310.0106560360896,
                    1303.9130794446796,
                    1297.1147768852381,
                    1296.5334044275544,
                    1285.2623795506088,
                    1280.2190240608948,
                    1271.5215103307041,
                    1263.6250746392034,
                    1253.5161758354761,
                    1251.5227219244803,
                    1239.0805493812816,
                    1233.4112544753434,
                    1228.538272682451,
                    1223.1731565372613,
                    1219.01418862669,
                    1210.7496591025274,
                    1200.6007095482298,
                    1191.2554550350721,
                    1189.6768756634244,
                    1182.7832211162513,
                    1175.9742311283294,
                    1167.8679764017534,
                    1158.056330265655,
                    1154.6265985477396,
                    1147.3203198481995,
                    1144.40083858406,
                    1127.2537476677041,
                    1122.8215679889888,
                    1118.6730538592415,
                    1113.762530074168,
                    1096.48342375024,
                    1094.6687851964562,
                    1086.5920728397275,
                    1079.7869121491992,
                    1076.1142408937317,
                    1070.0585403689379,
                    1065.9576302509277,
                    1049.999967838204,
                    1043.094585496335,
                    1036.2384248092092,
                    1023.8414979491517,
                    1019.8004762820856,
                    1005.828391886543,
                    1004.7662951058807,
            995.6767775707693,
            985.1038624262083,
            978.9889272026455,
            968.0499997361358,
            965.3420725252184,
            953.0023006526087,
            947.1854614707453,
            938.3278190782904,
            936.3984236601038,
            923.4609097029091,
            921.0794722574383,
            916.6788235846817,
            908.3403442332257,
            901.0615999651116,
            890.964231954489,
            877.3977247900167,
            876.0351417493059,
            871.9098264476945,
            858.1824216165412,
            851.9229557947579,
            847.1330806498127,
            842.5711057865219,
            823.2633036610592,
            820.0443942936928,
            805.2138404044144,
            802.5407977576783,
            793.4445148249908,
            785.2011158010483,
            782.1824295064267,
            777.6203782911355,
            769.5046294018971,
            762.3267018641719,
            754.0615402516937,
            749.8841899970478,
            740.986587075836,
            735.6691797188403,
            731.2280787672452,
            721.0164850326393,
            716.9551237549805,
            707.5866069806119,
            699.4662916402627,
            693.0277975860037,
            690.7698573302698,
            679.3488159760713,
            674.9973241716202,
            666.3742010250786,
            660.9944730232478,
            656.4158637836085,
            648.7467723482599,
            644.9005697601357,
            639.5658255899526,
            631.8187295952473,
            621.275345330835,
            615.6625755445742,
            609.2403349397488,
            603.8416948869592,
            598.5334699765037,
                    591.5977086644292,
                    588.8251788859022,
                    582.2123245464169,
                    574.9137586535109,
                    571.6122192725932,
                    568.935700618339,
                    560.4308830822855,
                    551.3056758644709,
                    547.744282257967,
                    539.1504107035821,
                    535.9212581474361,
                    526.6467533635199,
                    518.0002456557137,
                    515.6082537327087,
                    511.7684273780472,
                    507.9097090115244,
                    498.40739944476314,
                    496.5932780297685,
                    491.6406696658686,
                    487.0852055360594,
                    482.0994027523267,
                    478.15312169292986,
                    476.24569561420805,
                    472.7890879330935,
                    468.0565220541773,
                    464.93909837780967,
                    456.8091161301302,
                    452.20620337808475,
                    447.46992371698946,
                    444.9565363308063,
                    439.224336919205,
                    436.4522381885069,
                    431.4429012204554,
                    428.02685316840876,
                    425.4329999685212,
                    419.79479917525856,
                    412.1973960651148,
                    409.8443087275893,
                    408.5936818783972,
                    400.27481084905446,
                    398.49667232222276,
                    395.81504745209423,
                    389.27357858308324,
                    386.77813197155655,
                    383.1569383995079,
                    378.89760043925804,
                    376.01805746259924,
                    373.6839555386438,
                    368.8754837511592,
                    366.84225282102545,
                    362.857642259676,
                    356.9788586757997,
                    355.7430779985892,
                    353.7992674467541,
                    348.05890706705117,
                    345.64715239663406,
                    342.97016016877467,
                  338.66208597109863,
                  336.51151572906576,
                  334.8019871863371,
                  334.38023592931546,
                  327.9095607174337,
                  321.1772643181512,
                  320.23350625269984,
                  315.29105811252356,
                  311.26245119638327,
                  310.3806090428577,
                  304.3417947877482,
                  303.1820648113719,
                  299.73479167066006,
                  298.18628477168596,
                  296.3306124316166,
                  293.02432999011177,
                  290.2211809695783,
                  287.62766968268625,
                  285.3545899547165,
                  280.23512579763616,
                  278.7626329009849,
                  278.11892810068366,
                  275.92150030833915,
                  271.582377590406,
                  270.7444508896633,
                  267.2793831999976,
                  266.5202654063596,
                  264.8386769473172,
                  261.43861476583385,
                  259.71761622312704,
                  257.64945433982336,
                  255.3115779001129,
                  250.0631760228279,
                  247.61821202915212,
                  247.2442096361099,
                  246.71916029013934,
                  244.9591132467487,
                  242.5858646806605,
                  239.80497768889254,
                  238.65110873771684,
                  235.5616954465967,
                  234.65646006633804,
                  232.12650860400612,
                  228.8748311583666,
                  225.69787094768535,
                  225.10988984196885,
                  222.034121208587,
                  221.73943150617714,
                  219.7781569836461,
                  218.13564874097256,
                  215.4712366175431,
                  214.54486608072682,
                  212.10891396233637,
                  210.4614701152402,
                  209.10407051404144,
                  206.5339569607762,
                  205.65712018651624,
 203.7844652872573,
 200.85476783480578,
 198.18820868232336,
 197.61279032540716,
 197.45619077820498,
 195.54336283720443,
 193.76337517746072,
 191.24640756403815,
 190.7214802083437,
 189.184523225224,
 187.27341296667822,
 184.82324788071995,
 182.62259560731943,
 181.14206415181974,
 180.25437390825257,
 179.6642204669775,
 178.56070623102917,
 177.50115106369915,
 173.46352818460483,
 173.1121602558381,
 170.9169466988479,
 170.18085511470912,
 169.12674172527983,
 167.9159112689363,
 165.650870035845,
 165.09599936945256,
 163.2801895265787,
 163.15968268474904,
 160.13520978770183,
 160.08592974581018,
 157.88062221489085,
 157.12271302665482,
 156.19585937502097,
 155.3883967589895,
 153.65982712094547,
 152.76382132658492,
 151.72350244207624,
 150.85269542274605,
 150.75143940641036,
 148.86337457084664,
 147.47530651504024,
 146.64168624959285,
 144.89594441465806,
 143.79413833922567,
 142.79978192812314,
 141.76206221722916,
 141.4144443130969,
 140.06480146379337,
 137.48113868559523,
 137.1479461385645,
 135.1925708564781,
 134.70145965568412,
 134.08325461218553,
 132.1812927837395,
 131.78289520594714,
 131.16000268773848,
 129.9996335850959,
                  129.03251741381743,
                  128.40347253059304,
                  126.52387632933115,
                  126.05717048109526,
                  125.37315811176742,
                  124.75764669925795,
                  123.3616208760428,
                  122.19975067206686,
                  121.66707861633002,
                  120.617036163947,
                  119.99290714658265,
                  119.29765548531613,
                  119.05097958288695,
                  117.37726455362288,
                  116.1077833709493,
                  115.73635272147648,
                  114.6063607168675,
                  114.17912359786285,
                  113.0841137157251,
                  112.08791391780613,
                  111.58002237548448,
                  110.51178165557768,
                  110.18445814646547,
                  108.92188759900115,
                  107.19308716102249,
                  106.83919753973883,
                  105.56419345310445,
                  105.31404251952802,
                  104.40315505605777,
                  103.83964605493208,
                  102.96171715991046,
                  101.89139758041038,
                  101.47073290188864,
                  100.64489986703215,
                  99.92656945602691,
                  99.58058152706224,
                  99.03844760113178,
                  98.69163479024601,
                  96.53043557376107,
                  96.24721223022812,
                  95.42240029728131,
                  94.10168690750776,
                  94.01334069288639,
                  93.23843003297017,
                  92.48327692144368,
                  92.06038271239608,
                  90.89510071953953,
                  90.4142859354493,
                  89.54785845306668,
                  88.7238998421381,
                  88.39818077781534,
                  88.00284808208544,
                  86.8627735399565,
                  86.53345935175939,
                  86.11294917380663,
                  85.273917294425,
                  84.00062078629708,
83.81052592331953,
83.66597028222594,
82.65329424401945,
82.43444409288979,
81.64381223577904,
79.53332150777786,
78.91565442922975,
78.45406438181797,
77.88298302490945,
77.51612774916926,
77.1165341560877,
76.87618104026453,
75.83163087657995,
74.86463365649544,
74.362538067943,
73.92863450254771,
73.75308955461294,
73.0163281942214,
72.60258045215416,
71.60074969888132,
71.02346496815632,
70.63282370511952,
69.9037884313653,
69.43886953499359,
69.36147822079243,
68.64935010848136,
68.31906957804397,
67.35978639529843,
67.14765242610429,
66.75406778712089,
66.30067158841621,
65.69581744069816,
64.98363642834214,
64.41061414502529,
63.52010974132759,
63.237017541880235,
62.96668572783265,
62.359361155021986,
61.27871165007295,
61.16980004597682,
60.98267564415396,
59.90733489372986,
59.56549141291603,
58.947125797985116,
58.260718304026916,
58.09292868860934,
57.726476059160035,
57.34105712944193,
56.91485707511083,
56.77353517232039,
56.372606461583196,
55.62936883074075,
55.52440770112033,
54.8314668123557,
54.37659216666732,
54.14495921979941,
53.525169714000924,
                53.04101043077716,
                52.54753736324023,
                52.332115696277256,
                51.5622217395451,
                51.15961480527429,
                50.95025918093086,
                49.830003469550725,
                49.534887843058684,
                49.24408320825769,
                48.856902882517026,
                48.47555230403817,
                47.727406929036434,
                47.608238405821226,
                47.11821710257646,
                46.638827079629124,
                46.52533051841397,
                46.3234942390657,
                45.52782447102574,
                44.75145494896505,
                44.63070071308967,
                43.9246671199304,
                43.602884945686036,
                43.38285440095844,
                42.740187744357286,
                42.521197785145745,
                41.85968656637819,
                41.593414005362696,
                41.44296122204297,
                40.88320324866584,
                40.2495160343475,
                39.914567252876566,
                39.58465299220175,
                39.381365346974185,
                39.090053461071285,
                38.93761009764925,
                38.22962264730527,
                38.14663559757769,
                37.86636486029023,
                37.10778352413303,
                36.972951371668614,
                36.79122946273654,
                36.54099004453967,
                35.885063571378616,
                35.65120059393678,
                35.445703989064285,
                34.67947226759397,
                34.51687170637252,
                34.30427766378243,
                33.77609149617067,
                33.711024403410796,
                33.1490532530189,
                32.911263791269946,
                32.541147089656256,
                32.33681830647186,
                31.967642820594186,
                31.67890216406078,
                31.16908512784854,
          30.750393699242657,
          30.437566029655883,
          30.013349456294346,
          29.931842919977637,
          29.46984227881124,
          29.205559507004832,
          28.812335467272526,
          28.233505658696906,
          27.913557754665643,
          27.675558652985092,
          27.474110972187113,
          26.959631218172603,
          26.65518385784669,
          26.508222620453317,
          26.268415128105545,
          26.053621807896885,
          25.652613115925757,
          25.25826585855829,
          24.942688013559692,
          24.436366408185428,
          23.964874364990635,
          23.786783752757067,
          23.46419590006474,
          23.20519983466797,
          23.049355566834024,
          22.597944073846076,
          22.32864010475102,
          21.974110603090853,
          20.951002347958312,
          20.80205896938015,
          20.71649795986156,
          20.028278263024763,
          19.800234035882877,
          19.231505193390174,
          18.788019399453812,
          18.44442483340116,
          18.21446724558275,
          17.636102801826368,
          17.222825319527946,
          16.81919465218766,
          15.924247280540587,
          15.35935949762859,
          14.67113449421824,
          1.0829487406614121e-11,
          9.710799605921131e-12,
          9.45842070296644e-12,
          7.860238984524226e-12,
          7.505796108484484e-12,
          6.724154687170235e-12,
          6.3414286679422705e-12,
          5.85957286203495e-12,
          5.81002743232022e-12,
          5.4303930367523996e-12,
          5.332296732155085e-12,
          5.304403757278871e-12,
          4.820790543163779e-12,
          4.818716917015948e-12,
           4.393792529792439e-12,
           4.3797521421181605e-12,
           4.149064682209758e-12,
           3.826481049898039e-12,
           3.73079015594283e-12,
           3.706483166804645e-12,
           3.5231229653038286e-12,
           3.2708872940279544e-12,
           3.2644199400229795e-12,
           3.1703894332191306e-12,
           3.0558165227053176e-12,
           3.0231877094646596e-12,
           3.0041152541403984e-12,
           2.9933681809642884e-12,
           2.9861105714532967e-12,
           2.980585167666943e-12,
           2.9027936977472896e-12,
           2.8106618600680582e-12,
           2.739160528602449e-12,
           2.6862879718429395e-12,
           2.6543239090441353e-12,
           2.6369072511339466e-12,
           2.5843400951788807e-12,
           2.5180796267691534e-12,
           2.4796567179147995e-12,
           2.44909378493913e-12,
           2.4081175808025208e-12,
           2.399141354345797e-12,
           2.332616212143602e-12,
           2.3040241658454036e-12,
           2.3039773195206043e-12,
           2.2757645065347468e-12,
           2.27052291467369e-12,
           2.2323306556115024e-12,
           2.177025726606329e-12,
           2.164456113379587e-12,
           2.161032238553804e-12,
           2.136820872041382e-12,
           2.0860208168050854e-12,
           2.066134634811231e-12,
           2.0639481775585747e-12,
           2.0534009080017965e-12,
           2.0356953856711756e-12,
           2.018259080735976e-12,
           2.0065641864628236e-12,
           1.9893302705035166e-12,
           1.9525974798623806e-12,
           1.9399953266159268e-12,
           1.9383667820111272e-12,
           1.93711476324342e-12,
           1.9298528248792574e-12,
           1.9235532789414697e-12,
           1.9155764210613934e-12,
           1.8925865639591637e-12,
           1.8741607918447797e-12,
           1.8721026396616293e-12,
           1.8709821630430746e-12,
                 1.8579412919716673e-12,
                 1.856712404506013e-12,
                 1.8550291973808832e-12,
                 1.8441971224091674e-12,
                 1.8438014843236975e-12,
                 1.8144201082925055e-12,
                 1.7872551613209344e-12,
                 1.7619090331748408e-12,
                 1.755530525409195e-12,
                 1.755472559899534e-12,
                 1.7318621624379086e-12,
                 1.7268325935381583e-12,
                 1.7221871031990729e-12,
                 1.7168076502626639e-12,
                 1.6993769418073538e-12,
                 1.6867890047135667e-12,
                 1.6846603895271838e-12,
                 1.6840505770221492e-12,
                 1.6759300073102453e-12,
                 1.657838192718473e-12,
                 1.6400306549585037e-12,
                 1.6337610551070857e-12,
                 1.6322802155252733e-12,
                 1.6037518933810486e-12,
                 1.6035260045483506e-12,
                 1.5785603398520193e-12,
                 1.576608006566452e-12,
                 1.5743918480508747e-12,
                 1.5741133474751133e-12,
                 1.5664761986775967e-12,
                 1.5375387464069126e-12,
                 1.5349684569097611e-12,
                 1.5323353285668804e-12,
                 1.531798397019599e-12,
                 1.5265408787950404e-12,
                 1.5264076335496007e-12,
                 1.5237387104410501e-12,
                 1.5206015773610387e-12,
                 1.5114094441333102e-12,
                 1.4913417963074124e-12,
                 1.4776989733144116e-12,
                 1.4750603753169844e-12,
                 1.4741584433686568e-12,
                 1.4736624105768838e-12,
                 1.4721346132000613e-12,
                 1.4663229142013458e-12,
                 1.4507716930815234e-12,
                 1.443247741988117e-12,
                 1.4404641317726443e-12,
                 1.4357038490577032e-12,
                 1.4294590823741343e-12,
                 1.4261378196330385e-12,
                 1.4165493128056534e-12,
                 1.4158115882816832e-12,
                 1.4118348778680404e-12,
                 1.409446980527557e-12,
                 1.3980960635450975e-12,
                    1.3954196275439987e-12,
                    1.3874443368561405e-12,
                    1.3825271724568582e-12,
                    1.3813866655433446e-12,
                    1.3601005494407538e-12,
                    1.3397121407080445e-12,
                    1.327724067523146e-12,
                    1.3266807270610606e-12,
                    1.3208119609662183e-12,
                    1.3185317514420279e-12,
                    1.314875676605336e-12,
                    1.3131309789140375e-12,
                    1.3114011574984101e-12,
                    1.306687946696089e-12,
                    1.3055368101869008e-12,
                    1.3037214414439364e-12,
                    1.2990911365232697e-12,
                    1.2981067438290953e-12,
                    1.29073131090527e-12,
                    1.2846754005968267e-12,
                    1.284298156904591e-12,
                    1.2816956197934859e-12,
                    1.281156257174914e-12,
                    1.280766156540005e-12,
                    1.276282318485296e-12,
                    1.276175460077752e-12,
                    1.2742060835877096e-12,
                    1.2672900604163968e-12,
                    1.236758510461367e-12,
                    1.2367218764917498e-12,
                    1.2321899994871804e-12,
                    1.2250359981941047e-12,
                    1.2132498193433502e-12,
                    1.212961016141923e-12,
                    1.2039081021754896e-12,
                    1.1945460671630521e-12,
                    1.1910110403939388e-12,
                    1.1898775370294943e-12,
                    1.1894057493782603e-12,
                    1.178884365394856e-12,
                    1.1788218354954098e-12,
                    1.1762733818149219e-12,
                    1.1631312677138432e-12,
                    1.1550930390089202e-12,
                    1.1543727878210812e-12,
                    1.154069751657024e-12,
                    1.144629573591962e-12,
                    1.1374954776756257e-12,
                    1.1362791685603674e-12,
                    1.1358271227756364e-12,
                    1.1244777781279582e-12,
                    1.1132318051348643e-12,
                    1.1086531091969548e-12,
                    1.1024729412767931e-12,
                    1.1018744635473512e-12,
                    1.0957080200399377e-12,
                    1.0941125032700234e-12,
 1.0918034937586384e-12,
 1.0915575488333161e-12,
 1.0901492597358806e-12,
 1.0865860421677451e-12,
 1.0764199872619398e-12,
 1.0721814371761275e-12,
 1.0698255499331999e-12,
 1.0626324997995043e-12,
 1.0535547580989648e-12,
 1.0523107502098832e-12,
 1.0459088307414173e-12,
 1.0437685841306285e-12,
 1.0400215770429454e-12,
 1.0385531082574682e-12,
 1.0380868314992096e-12,
 1.0372050658652803e-12,
 1.0326985179315722e-12,
 1.0323689283489895e-12,
 1.0303913670705764e-12,
 1.029914573483078e-12,
 1.0238577716957089e-12,
 1.0211305820860468e-12,
 1.018718419179555e-12,
 1.0161293288409296e-12,
 1.013903875480287e-12,
 1.0090566097553101e-12,
 1.0036033811603256e-12,
 1.0019180326834833e-12,
 9.95624248252639e-13,
 9.935068941737739e-13,
 9.876760417336907e-13,
 ...]
```

In [90]:
```
# a = np.array(values_pca)
values_pca_diag=np.diag(flat_list)
```

In [91]:
```
values_pca_diag.shape
```

Out[91]:
```
(2066, 2066)
```

Projection

In [92]:
```
proj_dual=np.dot(values_pca_diag,v.T)
```

In [93]:
```
proj_dual=pd.DataFrame(proj_dual)
```

In [94]:
```
proj_dual.T
```

Out[94]:
```
                0             1             2             3             4              5
0     3260.281652   2076.393804    623.869540   1105.095014   2460.912277    1539.799577   11
1    -2710.230937  -1692.862034  -1063.848940  -1224.477082   -340.243865    1066.056449   -6
2     2270.480054   1280.691083    683.886484    766.714552    765.307684     163.901848   7
3    -2113.839617  -1134.958492    978.596760    779.579273  -1176.779627   -1098.110446   -3
4     1407.134811    519.842858   1784.060639   1068.151909   1545.517344    -159.517604   11
...           ...           ...           ...           ...           ...            ...
2061     0.000000   1200.214536   -358.633754  -1207.344120  -1968.416366   -2771.624279  110
2062     0.000000    132.742693   2150.474250  -7332.576765   6113.376989    5688.516573   11
2063     0.000000    763.482999   1776.728554  -7772.417624   3918.631772   -2723.270738  -43
2064     0.000000    451.044453  -2709.468130  -6163.260056  -4609.777451    4698.516903  -30
2065     0.000000  -7426.311757  -1104.710680   -668.809680  -7597.944130  -13038.582332  -15

2066 rows × 2066 columns
```

Reconstruction

In [95]:
```
x_hat_dual1=np.dot(v,df_features_znorm)
x_hat_dual=np.dot(v.T,x_hat_dual1)
```

In [96]:
```
x_hat_dual=pd.DataFrame(x_hat_dual)
```

In [97]:
```
x_hat_dual
```

Out[97]:

	0	1	2	3	4	5	6	7	
0	1.010077	0.966782	0.359594	-1.668004	-1.638671	1.007994	-0.324102	-0.992537	0.9
1	1.687176	-1.029924	1.026488	0.336317	-0.976018	0.340307	1.674690	-0.992537	0.9
2	-1.021220	0.301213	-1.641090	0.336317	-0.976018	-0.995067	-1.656630	-0.992537	-1.6
3	1.687176	0.301213	-0.307301	0.336317	1.674594	-0.327380	-0.324102	-1.648724	0.9
4	0.332978	1.632350	0.359594	0.336317	-1.638671	1.007994	-0.990366	-0.992537	0.9
...	...	...	...	...	...	...	...	...	
2061	1.010077	-1.695492	0.359594	-1.668004	1.011941	-1.662753	1.008426	0.319835	-1.0
2062	-0.344121	-0.364355	0.359594	1.004424	-0.313365	-0.995067	-0.324102	0.319835	0.3
2063	-0.344121	0.301213	-0.307301	0.336317	-0.976018	-0.327380	1.674690	1.632208	1.6
2064	1.687176	-0.364355	1.026488	0.336317	-0.976018	-1.662753	0.342162	-0.336351	-0.3
2065	0.332978	0.301213	-0.974195	0.336317	-0.313365	1.675681	1.008426	-0.336351	-0.3

2066 rows × 784 columns

In [98]:
```
stop = timeit.default_timer()
print('MODEL: ', stop - start)
Timetaken = Timetaken.append({'Model':'DUAL PCA', 'time':stop-start},ignore
_index=True)
```

```
MODEL:  5.822685161999999
```

In [99]:
```
Timetaken
```

Out[99]:

	Model	time
0	PCA	3.715777
1	DUAL PCA	5.822685

ANALYSIS::->

1.NOTE:-> Dual PCA is basically used when the number of dimensions >> no of observations to reduce the time of computations and save the computation storage as well. In our case case, Number of dimensions are less than the number of observations.

2. Execution Time:-> In our case, time taken by PCA is 3.71 seconds whereas time taken by the dual PCA is 5.8 seconds which is because of the fact defined in above NOTE. In an ideal case, Time and storage taken by the dual pca is always less compared to the PCA .

3.We tend to use the training samples into consideration while computing to reduce the computational time rather than taking the dimensions which are more in that particular case.

Citation:

1.http://www.math.uwaterloo.ca/~aghodsib/courses/f06stat890/notes/lec6.pdf

```
In [ ]:
```

2.2.2 Theoretical Question

Prove that PCA is the best linear method for reconstruction (with orthonormal bases). Hint: write down the optimization problem and solve it.

****Scanned and attached at end**

2.3.1 Fisher Discriminant Analysis (FDA)

```
In [100]: import timeit
          start = timeit.default_timer()
```

```
In [101]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
          x_train=df_features_znorm
          y_train=df_full['gnd']
          lda = LDA()
          lda_transformed1=lda.fit_transform(x_train, y_train)
```

```
In [102]: type(lda_transformed1)

Out[102]: numpy.ndarray
```

In [103]:
```
lda_tranformed1=pd.DataFrame(lda_transformed1)
```

In [104]:
```
type(lda_transformed1)
```

Out[104]:
```
numpy.ndarray
```

In [105]:
```
stop = timeit.default_timer()
print('MODEL: ', stop - start)
Timetaken = Timetaken.append({'Model':'LDA', 'time':stop-start},ignore_inde
x=True)
```

```
MODEL:  1.8334435830000047
```

In [106]:
```
lda_tranformed1
```

Out[106]:

	0	1	2	3
0	-5.277233	-2.052912	3.473823	-1.613518
1	-5.913727	-1.953482	3.665304	0.189606
2	-4.154543	-0.868888	1.172538	0.746605
3	-6.728769	-2.568941	4.442847	-0.640440
4	-6.977105	-2.125944	2.281978	-0.505802
...	...	...	...	...
2061	-1.493870	3.310943	-0.571532	0.611935
2062	-0.249322	4.925294	0.091210	-2.307546
2063	-1.311290	5.398086	0.216997	-1.333615
2064	0.213247	5.160965	-0.016662	-0.186851
2065	0.404523	4.317396	-0.313309	0.747472

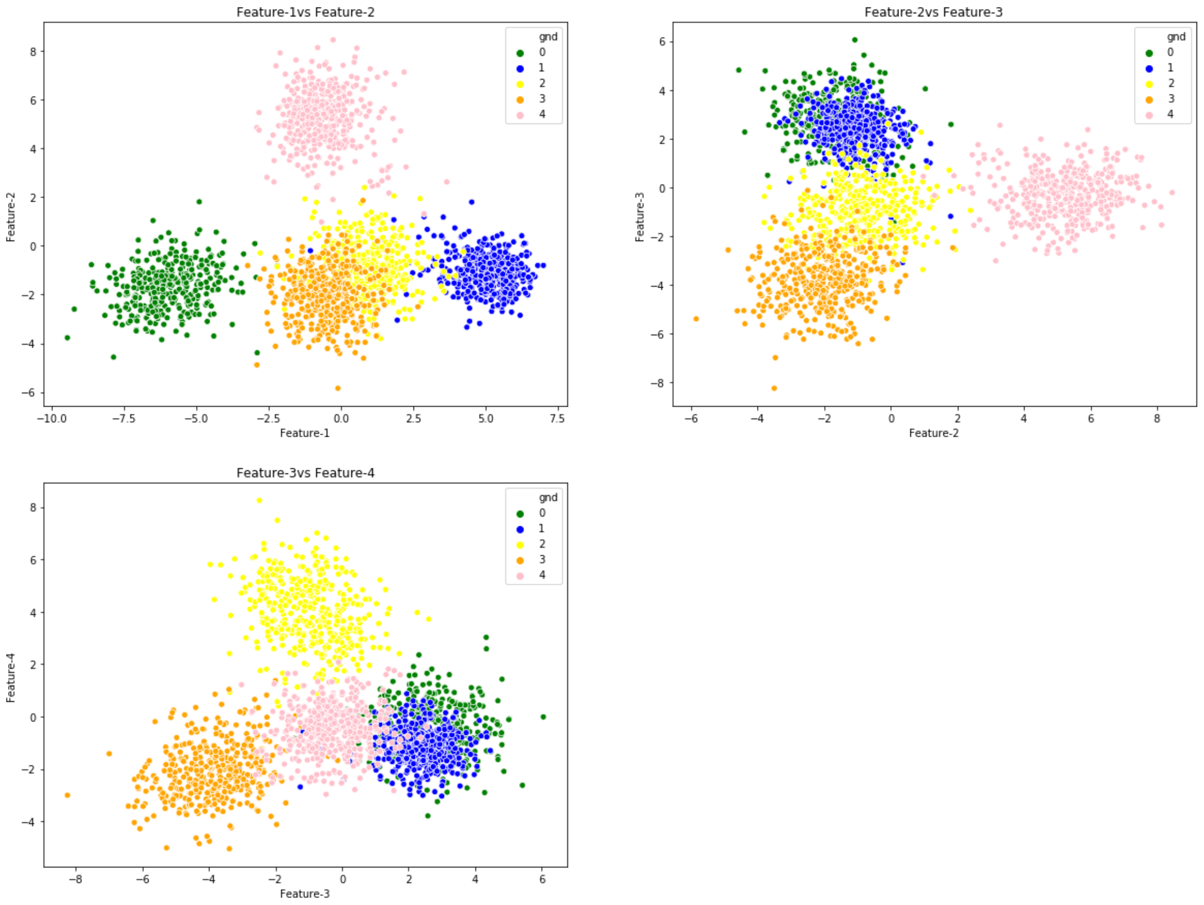
2066 rows × 4 columns

In [107]:
```
lda_with_label=pd.concat([lda_tranformed1,df_full.gnd],axis=1)
```

In [108]:
```
fig, axs = plt.subplots(2,2,figsize=(20,15))
k=0
fig.delaxes(axs[1][1])
for i in range(0,axs.shape[0]):
    for j in range(0,axs.shape[1]):
        sns.scatterplot(lda_with_label[k], lda_with_label[k+1],hue=lda_with_label.gnd,legend='full',ax=axs[i,j],palette=['green','blue','yellow','orange','pink'])
        axs[i,j].set(xlabel='Feature-'+str(k+1),ylabel='Feature-'+str(k+2))

        axs[i,j].set_title('Feature-'+str(k+1)+'vs Feature-'+str(k+2))

        if i==1 and j==0:
            break;
        k=k+1;
```



ANALYSIS::->

RESULTS [known classes] & SEPARABILITY=>

LDA1 VS LDA2

LDA1 :=>LDA 1 separates class label [0,1] from each other and from [2,3,4] very well.(i.e if we tend to draw a line on the LDA1 axes(x axes)) whereas Class labels [2,3,4] overlap with each other.

LDA2 :=>LDA 2 separates class label [4] from [0,1,2,3]very well if we draw a line on the LDA2 axes(y axes)). Class labels [0,1,2,3] overlap with each other.

LDA2 VS LDA3

LDA2 :=>LDA 2 separates class label [4] from [0,1,2,3]very well if we draw a line on the LDA2 axes(x axes)). Class labels [0,1,2,3] overlap with each other.

LDA3 :=>LDA 3 separates class label [3] from [0,1,2,4]very well if we draw a line on the LDA3 axes(y axes)). Class labels [0,1]and [2,4] overlap.

LDA3 VS LDA4

LDA3 :=>LDA 3 separates class label [3] from [0,1,2,4]very well if we draw a line on the LDA3 axes(x axes)). Class labels [0,1]and [2,4] overlap.

LDA4 :=>LDA 4 separates class label [2] from [0,1,3,4]very well if we draw a line on the LDA3 axes(y axes)). Class labels [0,1,3,4] overlap.

RESULT::=> It can be concluded that first direction of LDA separates classes better than the rest of the directions which can be theortically proved by finding the ranked eigen values.

COMPARISON BETWEEN LDA AND PCA RESULTS:==>

1.PCA lets us find the principal components with the highest variation in the data.

2.LDA finds the directions taking the variation between the classes and the variation within the class into consideration. i.e Maximize(Var_between) and Minimize(var_within)

3.From looking at the plots of both PCA and LDA, we can easily conclude that the "class separability" provided by the lDA directions is far better than that provided by the PCA, whereas the overall variation of data is best in PCA.

4.Correlation between the class and the direction is also more in case of LDA whereas PC1 and PC2 accounted for only 0.19 and 0.15 of correltion with the class labels.

Citations:=>

1.https://sebastianraschka.com/Articles/2014_python_lda.html

2.https://towardsdatascience.com/linear-discriminant-analysis-lda-101-using-r-6a97217a55a6

2.3.2 Theoretical Question

Scanned

Date

2.2.2 Theoretical Question:

PCA is the dimensionality reduction technique [linear].

Goal: (i) Preserve as much vas possible variance in the data in the new Coordinate System.

(ii) Reconstruct the original data from the projected data.

Let $X = [x_1, x_2 ----x_n]_{d \times n}$ be the datapoints. In order to approximate the space spanned by the X, we can choosep based on what amount of variance we would like to retain

To Proove: $X = [x_1 -- x_n]_{d \times n}$

$x_i \in \mathbb{R}^d$

d-dimensional vector.

Project these points on a vector U_1[PC]

$\underset{1 \times d}{U_1^T} \underset{d \times n}{X}$

$\Rightarrow$

We need to maximize $Var(U_1^T X)$

2nd maximum $Var(U_2^T X)$ and so on...

$\max_{u_1} Var(U_1^T X)$

Let $S \rightarrow$ be a $d \times d$ sample covariance matrix

$$\underset{1\times d}{U_1^T}\ \underset{d\times d}{S}\ \underset{d\times 1}{U_1}$$

$Var(U_1^T x) = U_1^T S U_1 \implies$ Can't take derivative $=0$ ∵ of no upper bound.

ie $\max(U_1^T S U_1)$

So after adding constraint, our optimization problem will be

OPTIMIZATION PROBLEM

$$\begin{cases} \max\limits_{U_1} (U_1^T S U_1) \\ \underline{\underline{s.t}}\ \ U_1^T U = 1 \end{cases}$$

| focus on: direction of highest variance first.

taking Lagrange.

$L(u_1, \lambda_1) = U_1^T S U_1 - \lambda(U_1^T U_1 - 1)$

↳ dual variable

This is dual form of optimization porblem

$L(U_1,\lambda_1) = U_1^T S U_1 - \lambda_1(U_1^T U_1 - 1)$

$$\frac{\partial L}{\partial U_1} = \cancel{2} S U_1 - \cancel{2}\lambda_1 U_1 = 0$$

$SU_1 = \lambda_1 U_1$ [matrix times a vector is scaler times a vector]

$S \to$ Covariance matrix

$U_1 \to$ Eigen vector of S } saddle pts of this Lagrange.

$\lambda_1 \to$ Eigen value of S } ie der$(L) = 0$

$\lambda_1 U_1 \ldots \lambda_n U_n \implies$ all are saddle pts.

which will maximize the variance

$U^T S U = U^T \lambda_1 U = \lambda U^T U = \lambda$ ie $\boxed{U^T S U = \lambda}$

Note: S has at most d eigen values & d eigen vectors.

for any U_n, value of obj. $f^n \implies \lambda_n$

Such that : $\lambda_1 > \lambda_2 > --- \lambda_n$ → Eigen Values
$U_1, U_2, --- U_n$ → Eigen vectors

Eigen Value: Amount of variance explained by each principal component.

Eigen Vector of Sample covariance matrix S Corresponding to the maximum eigen value is the first principal component.

We can find the Principal Components by the eigen value decomposition of the matrix S.

Now SVD,

$$\check{X} = (X_{d \times n} - M_{d \times n})$$
$$X = [x_1 —— x_n]$$
$$\check{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$
$$M = [\bar{x} \ \bar{x} \ . \ .]$$

$\check{X}\check{X}^T$ ⇒ Sample Cov matrix of X.

Now, if we do SVD of $\check{X}$, we get

$$\check{X} = U \Sigma V^T$$
$$\downarrow \qquad \searrow$$
$$\check{X}\check{X}^T \qquad \check{X}\check{X}$$

Column of U are eigen vectors of $\check{X}\check{X}^T$ Cov. matrix are principal components.

$[X \ U^T U)$

Recon,
$$Y = U_1^T X$$
$$y = U_1^T x_1$$

Date

Reconstruction :

$$\underset{\mathbb{R}}{\tilde{X}} = \underset{d\times 1}{U}\,\underset{1\times 1}{y}$$

$[\, y = U_1^T x_1 \rightarrow$ 1 pt. projected $]$

2.3.2

$$S_T = S_W + S_B$$

S_T : Total Scatter is the summation of within & b/w the scatters.

In PCA, Goal was to maximize the variance of the data whereas,

In LDA :

(i) Minimize the variance within each class.

(ii) Maximize the ~~variance~~ distance between the class labels

first class has to projected to W^T.

Projected mean $\Rightarrow W^T U_1$

$U_0 \rightarrow$ mean of class 1

$U_0 = \frac{1}{\text{no } y_i} \sum x_i$

Goal 2

$\max (W^T U_0 - W_1^T U_1)^2$.

$$L = [W^T U_0 - W^T U_1]^T (W^T U_0 - W^T U_1)$$

$$= (U_0 - U_1)^T (W)(W^T)(U_0 - U_1)$$

$$= W^T (U_0 - U_1)(U_0 - U_1)^T W$$

Date

W is unknown to us.

$(U_0 - U_1)(U_0 - U_1)^T \Rightarrow S_B$

If class 1 has Covariance matrix Σ_0

" 2 " " Σ_1

Class 1 has variance $W^T \Sigma_0 W$

class 2 has Variance $W^T \Sigma_1 W$

⇒ var of projected date.

Goal : minimize var ⇒ $W^T \Sigma_0 W$

find w ?

min $(W^T \Sigma_0 W + W^T \Sigma_1 W) \Rightarrow (\Sigma_0 + \Sigma_1) W$

S_W within class Covariance.

So maximize $(W^T S_B W)$

max $\dfrac{W^T S_B W}{W^T S_W W}$ ⇒ Optimization Problem.

equivalent to

max $W^T S_B W$

s.t. $W^T S_W W = 1$

This was the problem in PCA where we had to put constraint to make the optimization problem welldefined. will Consider length of w. and Search for the best direction. which was also the Case with PCA.

So, Applying the lagrangian

$$L(w,\lambda) = W^T S_B W - \lambda(W^T S w - 1)$$

On solving, we get.

$$\underset{\hookrightarrow \text{full rank}}{S_W^{-1}}\ \overset{\text{rank}=1}{S_B W} = \lambda W$$

So there is only 1 eigen Vector for 2dimensional data in LDA.

whereas, In PCA it was d dimensional.

LDA for multiclass problem can be explained by. :

j classes → j-1 dimensions

$W^T S_B W$ where $W = [w_1\ w_2 - w_{j-1})$

$\overset{a}{Tr(W^T S_B W)}$, $\overset{b}{Tr(W^T S_W W)}$

$$\max\left(\frac{Tr(a)}{Tr(b)}\right)$$

o.r

$$\max \left| \begin{array}{l} Tr(W^T S_B W) \\ Tr(W^T S_W W) = 1 \end{array}\right.$$

Apply lag.

$$\underbrace{\boxed{S_W^{-1} S_B}}_{j-1\ \text{rank}} W = \lambda W$$

$\therefore$ j-1 e.vectors

In $S_W = \Sigma_0 + ---\Sigma_j$

In 2

$$S_B = (U_0 - U_1)(U_0 - U_1)^T$$

Thus, $S_T = S_B + S_W$

where $S_T = \frac{1}{n-d} \sum_{i=1}^{n} (x_i - u)(x_i - u)$

$$\boxed{S_B = S_T - S_W.}$$

Theoritically, we have proved that LDA takes into account the distance b/w the classes as well in addition to the variance of date taken by PCA. which Can be clearly seen in the plots of [PC1, PC2] [PC2, PC3] and [LD1, LD2], [LD2, LD3] where

(1) LDA was able to separate the class labels very well and variance b/w whole date was also taken into aCcount.

(2) Whereas In PCA, whole date was widely spread with no clustering of the classes formed